International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Implementation of Efficient Architecture for Vulnarability Packet Detection Using Verilog

M. Sivaramprasad[1], D. Sridhar[2]

[1]P.G Student of Department of Electronics and Communication Engineering, SVIET, Nandamuru, JNTUK, Kakinada, Andhra Pradesh, India

[2]Assistant Professor of Department of Electronics and Communication Engineering, SVIET, Nandamuru, JNTUK, Kakinada, Andhra Pradesh, India

Abstract: ***Network security has always been an important issue and its application is ready to perform powerful pattern matching to protect against virus attacks, spam and Trojan horses. However, attacks such as spam, spyware, worms, viruses, and phishing target the application layer rather than the network layer. Therefore, traditional firewalls no longer provide enough protection. However, the solutions in the literature for firewalls are not scalable, and they do not address the difficulty of an antivirus. The goal is to provide a systematic virus detection hardware solution for network security for embedded systems. Instead of placing entire matching patterns on a chip, our solution is based on an antivirus processor that works as much of the filtering information as possible onto a chip. The infrequently accessing off-chip data to make the matching mechanism scalable to large pattern sets. In the first stage, the filtering engine can filter out more than93.1% of data as safe, using a merged shift table. Only 6.9% or less of potentially unsafe data must be precisely checked in the second stage by the exact-matching engine from off-chip memory. This gives a high efficiency, improved performance and high ability of packet detection with less contribution of time in an effective way***

Keywords: Algorithmic Attacks Embedded System, Memory Gap, Network Security, and Virus Detection.

1. Introduction

Network security has always been an chief issue. End users are vulnerable to virus attacks, spams and Trojan horses, for example. They may visit malicious websites or hackers may gain entry to their computers and use them as android computers to attack others. To ensure a secure network environment, firewalls were first introduced to block unauthorized Internet users from accessing resources in a private network by means of simply checking the sachet top (MAC address/IP address/port number). Over the past few years, there has been a substantial increase in the number of malware that have been in print for mobile devices. As per, there exist at least 31 families and 170 variants of branded mobile malware. Statistics have shown that at least 10 Trojans are released every week. Even however it took computer viruses twenty years to evolve, their mobile device counterparts have evolve for the duration of just a length of two years.

To understand the threat that is involved, we opening present the comparison of the environment used for PC-based and itinerant device malware. While dealing with a mammoth integer of virus this method drastically reduces the probability of creature attacked. nevertheless, attacks such when spam, spyware, worms, viruses, and phishing target the application sheet rather than the arrangement layer. Then traditional firewalls thumbs down longer provide enough protection. Many solutions, such as germ scanners, spam-mail filters, instantaneous messaging protectors, network shields, content filters, and peer-to-peer protector, have been in actual actuality implement. Initially, these solutions were position into service at the end-user side but be likely en route for be merged into routers/firewalls to provide profound protection. As a result, these routers stop threats on the network periphery along with keep them not in of corporate Networks. In this case, the firewall router might firstly deny some connections beginning the firewall based on the target's IP address and the connection port. Then, the fire-wall router would monitor the content of the web pages to prevent the user from accessing any page that connects to malware links or inappropriate content, based on content filters. When the user wants to download a compressed file, to ensure that the file is not infected, the firewall router must decompress this file and check it using anti-virus programs. In summary, firewall routers require several time-consuming steps to provide a secure connection. In some gear parallel combinational logic is applied at every one word in the memory and a test is completed next to the same time for coincidence with the search word. into other cases the search word and all of the words in the memory are shifted serially in synchronism; a single bit of the search expression is subsequently compared to the same bit of every solitary of the memory words using as loads of single-bit coincidence circuits as there are words in the memory. Amplifications of the associative reminiscence technique allow for masking the search word or requiring only a "close" amusement as opposed to an exact equivalent Small parallel associative memories are used in cache memory and effective recollection mapping applications. Cabir was developed for mobile phones running the Symbian and Series 60software, and using Bluetooth. The virus searches within Bluetooth's range (about 30 meters) for mobile phones running in discoverable mode and sends itself, disguised as a security file, to any vulnerable devices. The virus only becomes active if the recipient accepts the file and then installs it.

Once installed, the virus displays the word "Caribe" on the device's display. Each time an infected phone is turned on,

the virus launches itself and scans the area for other devices to send itself to. The scanning process is likely to drain the phone's batteries. Cabir can be thought of as a hybrid virus/worm: its mode of distribution qualifies it as a network worm, but it requires user interaction like a traditional virus .Since equivalent operations on many words are expensive (in hardware), a variety of stratagems are used on the road to approximate associative memory operation lacking actually carrying out the full test described here. solitary of these uses hashing to generate a "best speculation" for a conventional address followed bya test of the contents of that address. A data-storage device in which a location is identified by its informational content rather than by names, addresses, or relative positions, and from which the data may be retrieved.

2. Concept of a Firewall Router

Network firewalls and routers can use a rule database to decide which packets will be allowed from one network into another. By filtering packets the firewalls and router can imp rove security and performance - by excluding packets which may pose a security risk to a network or are not relevant to it. However, as the size of the rule list increases, it becomes difficult to maintain and validate the rules, and the cost of rule lookup may add significantly to latency. Ordered binary decision diagrams (BDDs)compact method of representing and manipulating Boolean expressions -- are a potential method of representing the rules.

This paper explores how BDDs can be used to develop methods that aid analysis of rules to validate them and changes to them, to improve performance, and facilitate hardware crutch up. 1 Introduction The growth of network and internet communication creates several challenges for network design. The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation (DEC) developed filter systems known as packet filter firewalls. This fairly basic system was the first generation of what is now a highly involved and technical internet security feature. At AT&T Bell Labs, Bill Cheswick and Steve Bellovin were continuing their research in packet filtering and developed a working model for their own company based on their original first generation architecture. Packet filters act by inspecting the "packets" which are transferred between computers on the Internet. If a packet matches the packet filter's set of filtering rules, the packet filter will drop (silently discard) the packet or reject it (discard it, and send "error responses" to the source). This type of packet filtering pays no attention to whether a packet is part of an existing stream of traffic (i.e. it stores no information on connection "state").Instead, it filters each packet based only on information contained in the packet itself (most commonly using a combination of the packet's source and destination address, its protocol, and, for TCP and UDP traffic, the port number).

TCP and UDP protocols constitute most communication over the Internet, and because TCP and UDP traffic by convention uses well known ports for particular types of traffic, a "stateless" packet filter can distinguish between, and thus control, those types of traffic (such as web browsing, remote printing, email transmission, file transfer), unless the machines on each side of the packet filter are both using the same non-standard ports. Packet filtering firewalls work mainly on the first three layers of the OSI reference model, which means most of the work is done between the network and physical layers, with a little bit of peeking into the transport layer to figure out source and destination port numbers.[9]When a packet originates from the sender and filters through a firewall, the device checks for matches to any of the packet filtering rules that are configured in the firewall and drops or rejects the packet accordingly.

When the packet passes through the firewall, it filters the packet on a protocol/port number basis (GSS). For example, if a rule in the firewall exists to block telnet access, then the firewall will block the TCP protocol for port number 23. Two imperative issues are safety and performance. When a new connection is established, the firewall router scans the connection and forwards these packet to the host after confirming that the connection is secure. Because firewall routers focus on the application layer of the OSI model, they must reassemble in-coming packet to restore the original connection and examine them through different application parsers to guarantee a secure set-up environment. For occurrence, believe a user search for information on web pages and then tries to download a com-pressed file beginning a web server.

In this case, the firewall router might initially deny some acquaintances from the firewall base on the target's IP address and the connection port. Then, the fire-wall router would monitor the content of the web pages to prevent the user from accessing any page that connects to malware links or inapt pleased, based on content filters. When the user wants to download a compressed file, to ensure that the file is not infected, the firewall router is obliged to decompress this file and check it using anti-virus programs.

3. Present System

There are many algorithms and accompanying hardware accelerators for fast pattern matching. One of the typical algorithms is the automation approach. This approach is based on Aho and Corasick's algorithm (AC), which introduces a linear-time algorithm for multi-pattern search with a large finite-state ma-chine. Its performance is not affected by the size of a given pattern set (the sum of all pattern lengths). In contrast, heuristic approaches are based on the Boyer-Moore algorithm, which was introduced in 1977.

Its key feature is the shift value, which shifts the algorithm's search window for multiple characters when it encounters a mismatch. However, attacks such as spam, spyware, worms, viruses, and phishing target the application layer rather than the network layer. Therefore, traditional

firewalls no longer provide enough protection. Many solutions, such as virus scanners, spam-mail filters, instant messaging protectors, network shields, content filters, and peer-to-peer protectors, have been effectively implemented. Initially, these solutions were implemented at the end-user side but tend to be merged into routers/firewalls to provide multi-layered protection.

As a result, these routers stop threats on the network edge and keep them out of corporate networks. The search window is a range of text exactly fetched by pattern matching algorithms for each examination. This algorithm performs better because it makes fewer comparisons than the naïve pattern-matching algorithm. At runtime, the Boyer-Moore algorithm uses a pattern pointer to locate a candidate position by assuming that a desired pattern exists at this position. The algorithm then shifts its search window to the right of this pattern. By default, desired patterns can exist in any position of a text; therefore, all positions in a text are candidate positions and must be examined. If the string of search windows does not appear in the pattern, the algorithm can shift the pattern pointer to the right and skip multiple characters from the candidate position to the end of the pattern without making comparisons.

Based on this concept, Wu and Manber (WM) modified the Boyer-Moore algorithm to search for multiple patterns. However, the performance of both of these algorithms is bounded by the pattern length. By default, desired patterns can exist in any position of a text; therefore, all positions in a text are candidate positions and must be examined. If the string of search windows does not appear in the pattern, the algorithm can shift the pattern pointer to the right and skip multiple characters from the candidate position to the end of the pattern without making comparisons. Based on this concept, Wu and Manber (WM) [18] modified the Boyer-Moore algorithm to search for multiple patterns. The WM algorithm is widely used in many applications, including UNIX tools such as agrep and glimpse. However, the performance of both of these algorithms is bounded by the pattern length. Its performance is not affected by the size of a given pattern set (the sum of all pattern lengths), but it requires a significant amount of memory due to state explosion. Experiments [17] have shown that the suboptimal AC algorithm requires 84.15 MB memory to represent Snort's rule set (4219 rules, as of December 2005). Even an Intel IXP2855 network processor (512kB on-chip memory) must store such a pattern set in off-chip memory.

Therefore, the memory hierarchy is the main factor in performance. Many previous studies have tried to lower memory requirements. In 2005, Lin Tan introduced a bit-split method by splitting an 8-bit character into four 2-bit characters to construct the automaton. Their state machines are smaller than the original, and they have fewer fan-out states for each transaction. However, the bit-split method reads several memory blocks in parallel when matching patterns. Thus, it can only be implemented by on-chip memory because of its high memory read port requirements. Piti Piyachon and Yan Luo extended this concept to the Intel IXP2855 network processor.

For increasingly large pattern sets, an IBM team implemented an optimized AC algorithm on the cell processor, and they discovered that the memory gap was the bottleneck. As a result, they modified the algorithm and used DMA to reduce the effect on the memory system. In contrast, heuristic approaches are based on the Boyer-Moore algorithm, which was introduced in 1977. Its key feature is the shift value, which shifts the algorithm's search window for multiple characters when it encounters a mismatch. The search window is a range of text exactly fetched by pattern matching algorithms for each examination. This algorithm performs better because it makes fewer comparisons than the naïve pattern-matching algorithm.

4. Virus Detection Processor

Focus on algorithms and have even developed for specialized circuits to increase the scanning speed. However, these works have not considered the interactions between algorithms and memory hierarchy. Because the number of attacks is increasing, pattern-matching processors require external memory to support an unlimited pattern set. This method makes the memory system the bottleneck. However, when the pattern set is already intractably large, a perfect solution is unattainable. Both engines have individual memories for storing significant information. For cost reasons, only a small amount of significant information regarding the patterns can be stored in the filtering Engine's on chip memory. In this case, we use a 32-kB on chip memory for the ClamAV virus database, which contained more than 30 000 virus codes and localized most of the computing inside the chip. Conversely, the exact-matching engine not only stores the entire pattern in external memory but also provides information to speed up the matching process. Our exact matching engine is space efficient and requires only four times the memory space of the original size pattern set. The size of a pattern set is the sum of the pattern length for each pattern in the given pattern set; in other words, it is the minimum size of the memory required to store the pattern set for the exact-matching engine. In this case, 8 MB of off chip memory was required for the ClamAV virus database (2 MB).

The filtering engine screens Impossible matches by consulting two TCAM lookup tables (named no plane and yes-plane), which are used to perform two steps of the on-chip data-scanning as shown in Fig1. Only important filtering signatures and skip data are stored on the chip. In order to reduce the on-chip memory, the filtering engine operates only on the fixed amount of the memory, including a 16-KB TCAM and a 8.5-KB SRAM.These filtering data are extracted from the entire virus database by pre-processing the 30K virus patterns released from the ClamAV. The operation principle of the virus-detection processor. The filtering engine screens impossible matches by consulting two TCAM lookup tables (named no-plane and yes plane), which are used to perform two steps of

the on-chip data-scanning.

The proposed exact-matching engine also supports data pre fetching and caching techniques to hide the access latency of the off-chip memory by allocating its data structure well. The other modules include a text buffer and a text pump that pre-fetches text in streaming method to overlap the matching progress and text reading. A load/store interface was used to support bandwidth sharing.This proposed architecture has six steps shown in Fig.2 for finding patterns.

Initially, a pattern pointer is assigned to point to the start of the given text at the filtering stage. Suppose the pattern matching processor examines the text from left to right. The filtering engine fetches a piece of text from the text buffer. If the position indicated by the pattern pointer is not a candidate position, then the filtering engine skips this piece of text and shifts the pattern pointer right multiple characters to continue to check the next position.

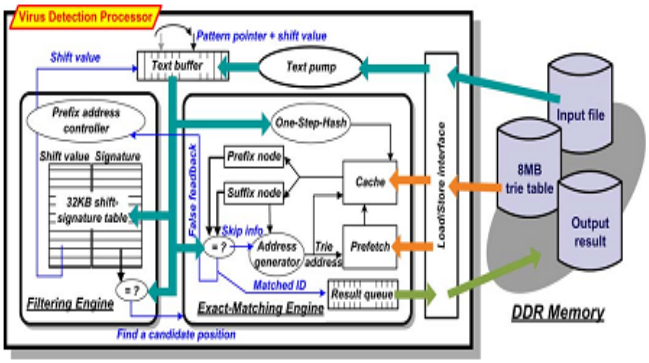


Figure 1: Virus Detection Processor Architecture

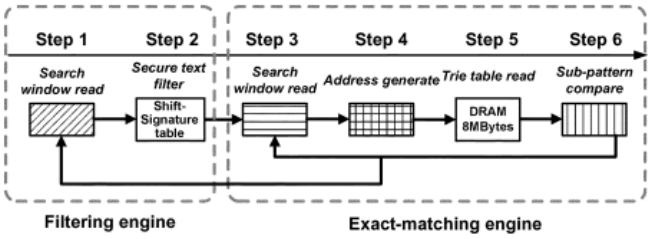


Figure 2: Two-phase pattern execution flow

Fig 2 Two-phase pattern execution flow conversely, the exact-matching engine not only stores the entire pattern in external memory but also provides information to speed up the matching process. Our exact-matching engine is space-efficient and requires only four times the memory space of the original size pattern set. The size of a pattern set is the sum of the pattern length for each pattern in the given pattern set; in other words, it is the minimum size of the memory required to store the pattern set for the exact-matching engine. In this case,8 MB of off-chip memory was required for the Clam AV virus database (2 MB). The proposed exact-matching engine also supports data pre fetching and caching techniques to hide the access latency of the off-chip memory by allocating its data structure well. The other modules include a text buffer and a text pump that pre fetches text in streaming method to overlap the matching progress and text reading. A load/store interface was used to support bandwidth sharing.

4.1 General Process

4.1.1 No-Plane Structure

The filtering engine screens impossible matches by consulting two TCAM lookup tables (named no-plane and yes-plane). Which are used to perform two steps of the on chip data-scanning to obtain a fast shift table. Which indicates the impossible matching patterns (so-called no plane). By comparing the input datum with the no-plane TCAM from the least significant bit (LSB), the engine first looks up the shift table to perform a quick shift of impossible bytes until locating a possible match. If the input datum is matched with an entry of no-plane, the input string will be skipped according to the shift count stored in the shift SRAM

4.1.2 Yes plane Structure

When the comparison of no-plane is missed or if the corresponding shift-count is zero, the filtering engine will enter the second step of virus detection. Then we further look up another signature table (called the yes-plane) to eliminate any false positives by ensuring that the prefix has the same signature. The filtering engine will skip the input datum if it is mismatched with the data of the yes-plane. If a possible match is still not ruled out, then the exactly-matching engine performs suffix matching by making comparisons with a suffix tree stored in off-chip memory, which can hold a large number of virus patterns. The yes-plane TCAM to reduce more exact comparisons. The filtering engine will skip the input datum if it is mismatched with the data of the yes-plane. If a possible match is still not ruled out, then the exactly matching engine performs suffix matching by making comparisons with a suffix tree stored in off-chip memory, which can hold a large number of virus patterns. The off chip memory needs roughly 8MB to store the entire 2MB virus patterns of the ClamAV .Our idea is to merge these two single-port TCAMs into a single rectangular dual-port TCAM and concurrently match with the whole prefix. To achieve this goal we need a dual-port TCAM and two SRAMs as shown in the right part of FIG, with a division line inserted in the dual-port TCAM array to separate the no-plane entries and the yes-plane entries. With the proposed dual-port TCAM, the ternary cells storing “X” terms can be minimized, and consequently both the total memory capacity and the power consumption are reduced It includes two single-port TCAMs and two SRAMs. One TCAM serves as the no-plane.

4.2 Wu-Manber Algorithm

The Wu-Manber algorithm is a high performance, multi pattern matching algorithm based on the Boyer-Moore algorithm. It builds three tables in the pre processing stage: a shift table, a hash table and a prefix table. The Wu-Manber algorithm is an exact-matching algorithm, but its shift table is an efficient filtering structure. The shift table is an extension of the bad-character concept in the Boyer-Moore algorithm, but they are not identical. The fig 3 shows Wu-Manber Algorithm match flow.

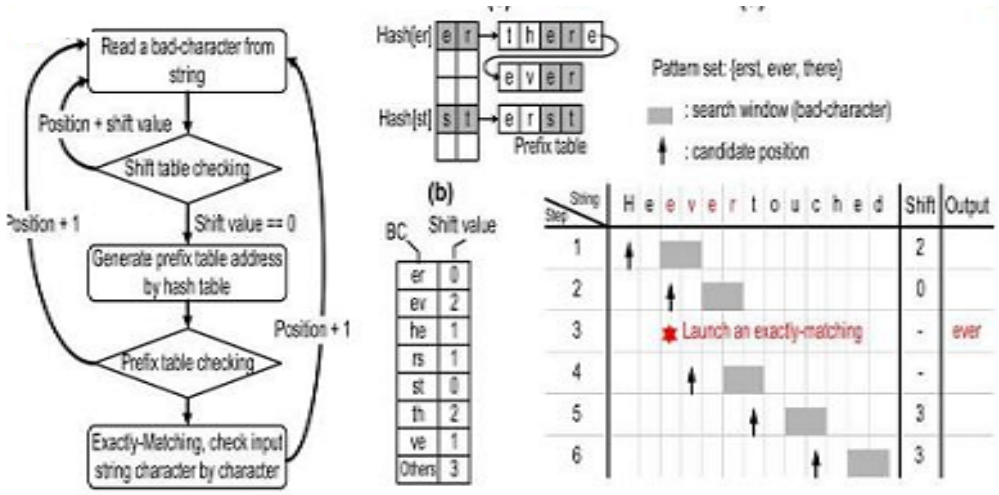


Figure 3: Wu-Manber matching process. (a) Matching flow; (b) shift table;(c) hash table + prefix table; (d) matching process.

4.3 Bloom Filter Algorithm

A Bloom filter is a space-efficient data structure used to test whether an element exists in a given set. This algorithm is composed of different hash functions and a long vector of bits. Initially, all bits are set to 0 at the pre processing stage. To add an element, the Bloom filter hashes the element by these hash functions and gets positions of its vector. The Bloom filter then sets the bits at these positions to 1. The value of a vector that only contains an element is called the signature of an element. To check the membership of a particular element, the Bloom filter hashes this element by the same hash functions at run time, and it also generates positions of the vector. The fig 4 shows bloom filter algorithm match flow. Fig 4 Matching flow

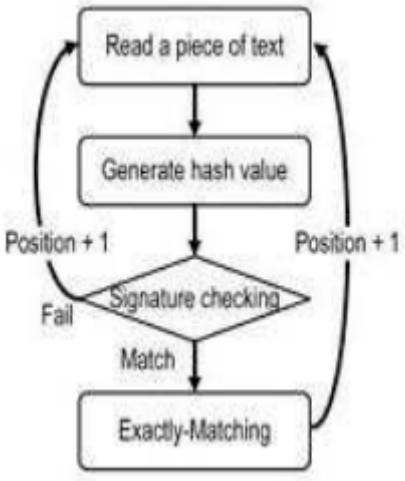


Figure 4: Matching flow

The filter only hashes all of the pattern prefixes at the preprocessing stage. Multiple patterns setting the same position of the bit vector are allowed. The arrows indicate the candidate positions. The gray bars represent the search window that the Bloom filter actually fetches for comparison. Both the candidate position and search window are aligned together. Thus, the Bloom filter scans and compares patterns from the head rather than the tail, like the WuManber algorithm. In step1, the filter hashes "He" and mismatches the signature with the bit vector. The filter then shifts right 1 character and finds the next candidate position. For the search window "ee", the Bloom filter matches the signature and then causes a false alarm to perform an exact-matching in steps 2 and 3.

The filter then returns to the filtering stage and shifts one character to the right in step 4, which launches a true alarm for the pattern "ever". Finally, the Bloom filter filters the rest of text and finds nothing. The Bloom filter then sets the bits at these positions to 1. The value of a vector that only contains an element is called the signature of an element. To check the membership of a particular element, the Bloom filter hashes this element by the same hash functions at run time, and it also generates positions of the vector. If all of these bits are set to 1, this query is claimed to be positive, otherwise it is claimed to be negative.

The output of the Bloom filter can be a false positive but never a false negative. Therefore, some pattern matching algorithms based on the Bloom filter must operate with an extra exact-matching algorithm. However, the Bloom filter still features the following advantages: 1) it is a space-efficient data structure; 2) the computing time of the Bloom filter is scaled linearly with the number of patterns; and 3) the Bloom filter is independent. 4.4 Shift-Signature Algorithm The proposed algorithm re-encodes the shift table to merge the signature table into a new table named the shift-signature table. The shift-signature table has the same size as the original shift table, as its width and length are the same seeing that the original change counter. There are two fields, S flag with carry, in the shift signature table.

The carry meadow has two types of data: a shift value and a signature. These two data types are used by two different algorithms. Thus, the S-flag is worn to designate the data type of a carry. The filtering steam engine can then filter the text using a different algorithm at the same time as providing a higher filter rate. The system used to merge these two

tables is described as follows. First, the algorithm generates two tables, a alter table and signature table, at the pre processing period. The age bracket of the shift table is the same as in the Wu-Manber algorithm. The S-flag is a1-bit field used to indicate the data type of the bring Two data types, shift value or signature, are defined for a carry.

The size and breadth of the shift signature counter are the same as those of the original shift table. To join these two tables the algorithm maps both entry in the shift table and autograph table onto the shift signature table. For the non-zero shift values, the S-flags are set, and their original shift values are cut out at 1-bit to fit their carries. Conversely, for the zero change values, their S flags are clear, and their carries are used to store their signatures. In this method, all of the entries in the shift-signature table contribute to the filtering rate at run time. Because of the address collision of bad characters, most entries contain less than half of the maximum shift distance for a large pattern set. Therefore, although this method sacrifices the maximum shift distance, the filter rate is not reduced but rather improved. The fig 5 shows Shift-Signature Algorithm match flow.

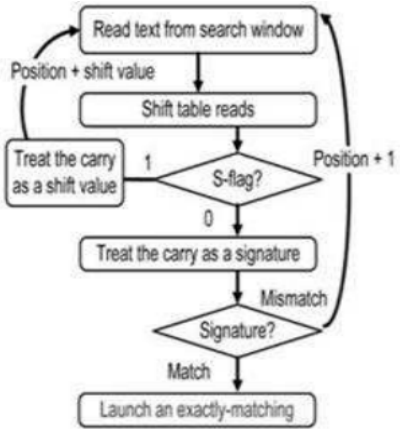
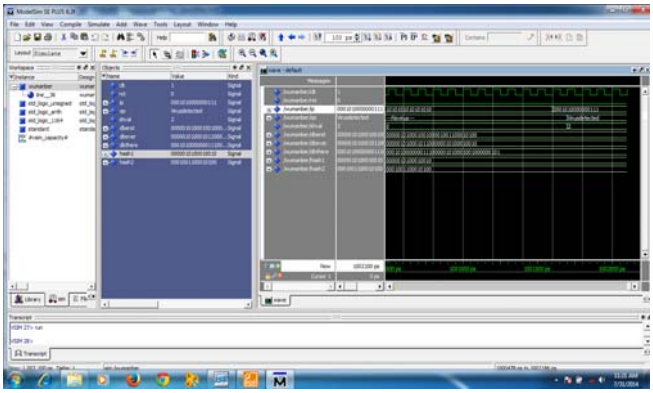


Figure 5: Matching flow

5. Result & Analysis

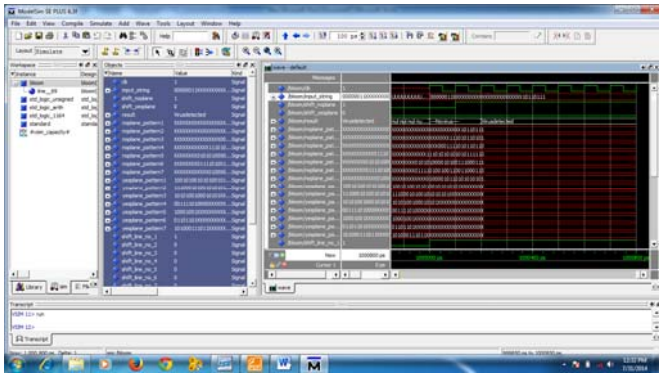
5.1 Simulation Reports

Wu-Manber Algorithm



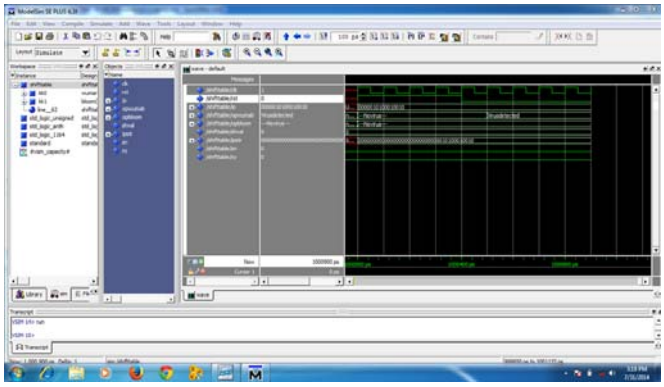
Inference: clk=clk, rst=1, after run you will get **no virus** after make rst=0, apply 15 bit input string =ip if the iput string is virus pattern set values i.e, is ip="0001001100010100" then –output was **virus detected**

5.2 Bloom Filter Algorithm



Inference:clk=clkandinput string=00000011XXXXXXXXXXXXXXXXXXXXXXX 10110111(underlined data is signature value i.e if not present in that signature it will go to next plane in that way we will save the time of execution) and non underline data is pattern
Then –output was **virus detected**

5.3 Shift signature Table

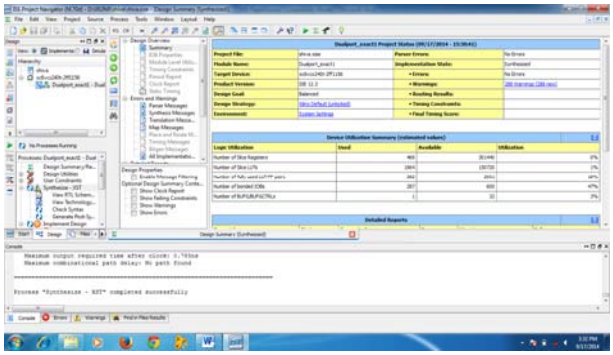


Generallyshift table can be constructed from the Combination of wumanber algorithm and bloom filter **Inference:**clk=clk, and rst =1 after run rst=0,apply

ip=0000010100010010 output was **virus detected.**

5.4 Synthesis Report

Here we increased number of virus patterns compared to many previous designs include claimed to make available high performance,



6. Conclusion

In this paper we describe a novel architecture for prototype matching virus detection processor for network intrusion unearthing system. The virus detection -processor is RAM-based aim which be used to store the additional bug model to hit upon the virus patterns .the dual port morsel CAM be

Dexterous pattern matching train is accomplished of detect added big patterns. Since the pattern are mechanical hooked on the co-processor with software, the planning can continue to exist used to implement design in FPGA as fighting vigorous as ASIC We have shown with the intention of our blueprint filter survive talented of yielding concert that surpass the most recent FPGA implementations while enabling the users to course it without having to regenerate moreover reconfigure the hardware. Such quick configuration may become critical, as the rate of coming out of new attack increase. Many previous designs include claimed to make available high performance, but the memory gap created by using external memory decrease recital because of the increasing size of virus databases. Furthermore, imperfect resources restrict the expediency of these algorithms used for embedded network security systems. Two-phase heuristic algorithms are a solution with a tradeoff between performances and cost due to an efficient filter table accessible in internal recollection however, their performance is without problems threatened by malicious attacks. This work analyzes two scenarios of malevolent attacks and provides two methods. The design of the adjustable division line provides high flexibility for updating virus databases.

References

[1] TSMC 0.13µm Logic 1P8M Salicide CU FSG 1.2V/3.3V Process Documents, Taiwan Semiconductor Manufacturing Co., Ltd.
[2] F. Yu, R. H. Katz, and T. V. Lakshman, “Gigabit rate packet pattern matching using TCAM,” in Proc. 12th IEEE Int. Conf. Netw. Protocols, 2004, pp. 174–178.intrusion detection system,” ACMTrans. Embed. Compute. Syst., vol. 3, pp. 614–633, 2004.
[3] D. P. Scarpazza, O. Villa, and F. Petrini, “High-speed string searching against large dictionaries on the Cell/B.E. Processor,” in Proc. IEEE Int. Symp. Parallel Distrib. Process. 2008, pp. 1–8.
[4] S. Dharmapurikar, P. Krishnamurthy, and T. S. Sproull, “Deep packet inspection using parallel bloom filters,” IEEE Micro, vol. 24, no. 1, pp.52–61, Jan. 2004.
[5] L. Tan and T. Sherwood, “A high throughput string matching architecture for intrusion detection and prevention,”in Proc. 32 nd Annu. Int. Symp. Comput. Arch., 2005, pp. 112–122.
[6] Chieh-Jen Cheng, Chao-Ching Wang, WeiChun Ku, Tien-Fu Chen, and Jinn-Shyan Wang, “Scalable High-Performance Virus Detection Processor Against a Large Pattern Set for Embedded Network Security” Commun. VOL. 20, NO. 5, MAY 2012.
[7] V. Aho and M. J. Corasick, “Efficient string matching: An aid to bibliographic search,” Commun. ACM, vol. 18, pp. 333–340, 1975.
[8] O. Villa, D. P. Scarpazza, and F. Petrini, “ Accelerating real-time string searching with multicore processors,” Computer, vol. 41, pp. 42–50,2008.
[9] R.-T. Liu, N.-F. Huang, C.-N. Kao, and C.-H. Chen, “A fast string matching algorithm for network processor-based intrusion detection system,” ACMTrans. Embed. Comput. Syst., vol. 3, pp. 614–633, 2004.
[10] Micron Technology, Inc., Boise, ID, “256MB DDR2 SDRAM datasheet,” 2003.

Author Profile

M. Sivaramprasad doing M.Tech VLSI System Design in Sri Vasavi Institute of Engineering and Technology, Nandamuru, Received B.Tech degree in Electronics and communication Engineering at Sri Vasavi Institute of Engineering and Technology, Nandamuru.

D. Sridhar Received the M.Tech degree in VLSI System Design from Avanthi Institute of Engineering and Technology, Narsipatnam, B.Tech degree in Electronics and communication engineering at Gudlavalleru. He has total Teaching Experience (UG and PG) of 7 years. He has guided and co-guided 5 P.G and U.G students. His research areas included VLSI system Design, Digital signal processing, Embedded systems