University of Essex

Design for Reconfigurable Embedded Security Applications

Department of Electronic Systems Engineering

Kevin Cheng and Martin Fleury
Department of Electronic Systems Engineering,
University of Essex, Colchester CO4 3SQ, UK.
Email: {kcsche;fleum}@essex.ac.uk

Abstract

Denial-of-service attacks are an increasing problem in today's networks. Embedded security systems are required as purely software defences are unable to cope with packet rates on high-speed networks. Reconfigurable logic is well suited to the changing nature of the threat. What is required is a general packet-scanning structure that can be applied to a range of development boards and target systems. Multiple clock domains allow the network interface to be de-coupled from the security logic. Hardware compilation on a platform FPGA enables changing threats to be quickly met, trading speed of design against optimality. Automatic re-timing offers a way to optimize clock widths. This poster presents some general design issues.

1. Introduction

Embedded network security systems are on the increase as purely software approaches cannot cope with existing packet throughputs, let alone high-performance LANs such as Gb Ethernet. For example, Sourcefire Inc., well known for Snort — a rule-based software search engine, have turned to an intrusion detection system using up to ten G5 PowerPC processors aimed at fibre-optic line-rates of 2–8 Gbps. SourceFire prefer not to use an ASIC, because these are not adaptable to new exploits. A look-up-table-based (SRAM) FPGA is reconfigurable and also supports greater throughput than a RISC, as it does not suffer from the fetch-execute bottleneck and can process multiple parallel streams according to need. As a comparison, the AES encryption algorithm runs at 1.5 Gbps on a Pentium 4 (3.2 GHz), at 12.2 Gbps on a Virtex XCV1000 FPGA, and 25.6 Gbps on an Amphion ASIC at 200 MHz clock speed. Hardware security devices also bring reduced vulnerability to attacks against the device itself (from software afflictions such as virus, worms, and executable content) and can act as an insulating layer around a PC and its data. A packet scanner provides passive defence at a home or campus PC. Packet scanning also has a role in protection of routers in the network core. Packet scanning differs from firewall protection in that it can act against the packet payload and that stateful filters can be implemented, i.e. filters that are responsive to patterns of activity over time.

2. Packet Scanners

Packet scanners work in reactive mode, acting to thwart newly discovered exploits. Therefore, a hardware system should also be able to quickly adjust its response. Reconfigurable hardware in the form of a SRAM FPGA is well suited to this task but only if it is also possible to quickly re-program the array. Fortunately, from the point that a netlist is available, the process of place-and-route is largely automated, unless optimised designs are required. There are three higher-level ways to arrive at a netlist: 1) by means of a hardware description language (HDL); 2) through a silicon compiler; or 3) using a hardware compiler. HDLs have the disadvantage that compilation times for large designs are lengthy, slowing down design iterations, though the range of associated tools allows low-level optimisations. A silicon compiler gives a succinct circuit description, often in an existing software language. A hardware compiler, such as Handel-C, converts a program or more accurately an algorithm into hardware. Hence, a hardware compiler exists at a higher level of abstraction allowing faster production of attack detection routines at a cost in control of the form of the output circuit. Since platform-FPGAs have become available from the two main manufacturers, Xilinx and Altera, gate (or rather slice) usage has become less critical, and certainly is not an issue for packet scanning routines.

3. Suspicious Content

Suspicious content is identified in a number of ways. Signatures (unique byte sequences) can be matched against a packets content (either header or payload) or conversely a hash of successive segments of a packets content is matched against a database of such hashes. The signatures or hashes can be stored either in external SRAM banks, in block RAM on the FPGA as conventional arrays or as Content-Addressable Memory (CAM). We have examined on-chip FPGA CAMs but a memory hierarchy, combining block RAM and SRAM, is also possible. Particularly in denial-of-service attacks, it is possible for an exploit to extend over a number of packets, requiring stateful filter structures such as counters and timers. Again, these can be pre-designed components of a generic structure. There are a variety of evaluation and development boards, such as the RC200/300 range from Celoxica Ltd, Tararis content processor, Xilinxs ML300, Digilent Inc.s XUP V2P, along with production boards, which implies that a structure that will also extend to future boards should be sought, not least because it will allow exchange of security routines or filters.

4. Hardware String Matching

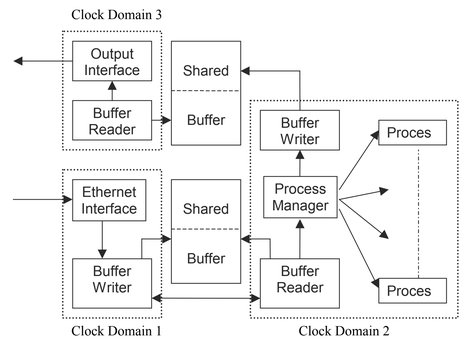
There are at least five approaches to hardware string matching: 1) String matching as in the software approach using Boyer-Moore, Aho-Corasick and similar algorithms; 2) non-deterministic finite automata (NFA); 3) comparators; 4) hashing, which involves approximate matching and, hence, can lead to false positives; 5) combinations of the others. The following designs in the literature have all been implemented on Virtex FPGA. Using approach 1), the well-known Knuth-Morris-Pratt (KMP) algorithm was employed to search a character at a time, matching by comparators. The main advantage of the KMP is said to be the ability to scale more readily, i.e. apply multiple rules by repeated application of the KMP, i.e. through pipelining. Approach 2) was applied to regular expressions. Approach 3 has been applied in which characters from a single packet are fanned out to a set of comparators, which is similar to the operation of a CAM. a CAM. Approach 4) was applied using a Bloom filter to match multiple strings at the same time. For each string, multiple hash functions are applied, each function outputting one value from a finite set of values. The resulting bit pattern vector acts as the search string. Bloom filter matching only identifies candidate threats, implying that a further exact match is required if false matches are to be avoided. Multiple Bloom filters are possible, acting in parallel on the same packet. This implies that variable length IP packets must be buffered. Thus, higher memory usage is a disadvantage of this approach. A similar comment applies to packet-wise parallelism in which a dispatcher places an arriving packet in one of four content-scanners (regular expressions). On the other hand, logic usage grows according to the number of characters in character-based approaches, whereas this is not the case for hashing methods.

5. Structures

Threat response time can also be improved if a pre-existing structure exists into which a scanning routine can be slotted. Device driver libraries have become available, such as Celoxicas PAL/PSL library, to allow an FPGA to interface to Ethernet and PCI busses, as well as standard computer peripherals such as RS-233, PS-2 and VGA. Network device interfaces are clearly of particular importance for a packet scanner.

Beyond that a buffering structure is needed that will allow a set of scanning routines to work in parallel on one packet, while allowing arriving packets to be stored in a custom data-structure. Access to the buffer must be regulated in hardware to prevent over-write. This is a different problem to software concurrency control through software semaphores or monitors, as those solutions assume virtual concurrency simulated by an operating system scheduler. A generic buffering structure will meet the needs of a variety of scanning routines.

6. Example Design



Technology mapper	Retiming	Virtex-II 5120 slices	Min. clock width *Domain 2 only*
No	No	13 %	12.610 ns
Yes	No	13 %	11.236 ns
Yes	Yes	14 %	10.936 ns

Partition of the design into separate clock designs did bring gains in relative clock speed in a Handel-C implementation, as clock domain 2 now is approximately 100 MHz, whereas the same code ran at 50 MHz in the one clock domain structure. Partitioning is a useful strategy in terms of generic designs with differing I/O interfaces and interface code. Currently, though packet throughput would be around 100 M packet/s with a single process as the input interface reduces this by a considerable amount. A shared buffer has been deployed against packet headers, which are fixed in width. A weakness of shared buffer approaches in general is that if the data payload is searched then the buffer slots must be of variable width. Dividing the payload into equal-sized chunks may solve this problem but the approach has still to be verified. Clock timings will be the main determinant of relative performance in comparison with a Snort-like software approach. A testing structure has already been constructed so that a stream of packets can be directed towards Snort and towards a packet scanner. Of course, this is on an isolated network. In fact, testing is not simple, as the software access to the raw socket interface is required to modify packet headers. Regulating access to a buffer by limiting the reader to a single clock cycle has a potential to increase the clock width if the reader is augmented in any way. Automatic re-timing offers some help in that respect. Minimizing the application clock domain clock width and the running speed is required for long running processes, so as to match packet arrival times. Hence, design of embedded systems for network security is a challenging task, which brings with it a range of new problems.

Poster presented at: *U.K. Design Forum Workshop, Manchester, April 2006*