

基于 SOPC 的网络入侵检测中模式匹配系统设计

摘要： 模式匹配作为网络安全的核心技术与实现难点越来越受人们关注。本文设计了一种基于 FPGA 的模式匹配系统，用 Verilog HDL 语言实现系统主体；采用开源的 Snort 规则，选用由异或运算组成的适合 FPGA 处理的 HashMem 函数进行模式匹配；通过软件预处理找出 Snort 中的冲突模式串进行单独匹配从而用硬件方法解决冲突。硬件电路上用 DM9000A 网络控制器接收网络数据。实验结果显示，本文吞吐量达到了 61Mbps，为 PC 上运行的软件方法的 2 倍以上；每百兆吞吐量为 61 Mbps/百兆，远远高于软件方法的 2.06 到 3.71 Mbps/百兆。当处理的 Snort 规则数增多时，系统资源消耗增加很少，吞吐量不受影响，与传统系统随着规则数增加性能下降相比更具优势。

关键词： 模式匹配；入侵检测；FPGA；Snort；HashMem 算法

Design of NIDS Pattern Matching System Based on SOPC

Abstract: Pattern matching as an important technology of network security has been getting more and more attention recent years. This paper presents a pattern matching system that based on FPGA. This paper uses DM9000A to receive network data and uses Snort rule and HashMem function to match pattern. With software simulation the conflict pattern string of Snort rules found out and processed separately. So, conflict can be high-speed solved. The experimental results show that the system throughput is 61Mbps, more than 2 times of the software method. When processing more Snort rules, the system resource consumption increased very little, throughput is not affected, which is more advantage compared to the performance of traditional systems.

Keyword: pattern matching; intrusion detection; FPGA; Snort; HashMem Algorithm

1 引言

入侵检测系统作为一种主动的安全防护手段，在计算机网络安全中扮演着重要的角色。近年来，网络规模和网络带宽的不断扩大，传统的入侵检测算法已经很难适应系统对高速、海量数据的处理需求。而作为入侵检测中关键部分--模式匹配模块，它的处理性能决定了整个入侵检测的性能。

为了提高模式匹配速度，人们分别从改进软件算法和应用硬件加速技术两个方面进行了相关的研究。目前，软件实现算法主要包括：Rabin-Karp 算法，Aho-Corasick 算法和 Boyer-Moore 算法等^[1]。硬件加速技术主要通过设计专用集成电路的方式来实现。

由于本身的限制，这两个方向的研究都没有使模式匹配技术的性能得到较好的提升。基于软件方式的模式匹配技术受处理器性能与软件串行执行等因素影响，处理速度的提升有限；基于专用集成电路的模式匹配技术虽然可以实现高速匹配，但是目前网络安全的检测规则变更较快，由于专用硬件电路不可以改变，所以无法适应检测规则的频繁更新。本文提出应用 FPGA 技术来实现模式匹配，通过在 FPGA 芯片中使用硬件描述语言，设计实现逻辑，经综合后，形成硬件电路，以类似专用硬件电路的方式实现模式匹配，极大的提高了处理性能，同时将模式匹配算法设计成多路输入方式，进而实现了多路数据的并行处理，提高了数据处理的吞吐量。针对检测规则的频繁更新，在 FPGA 芯片中，针对更新内容，在原有逻辑上进行规则的添加就可以满足更新的需求。

目前，一些学者也进行了基于 FPGA 的模式匹配方法的研究。第一类主要采用 FSM，NFA 和 DFA 等处理模式进行匹配^[2, 3]。但是，此类算法消耗大量的逻辑单元，并且，随着模式串数量增多，其消耗的硬件资源也会大幅增加，因而很难在普通 FPGA 器件上实现并普及。第二类采用 CAM、TCAM 实现模式匹配^[4]。用 CAM、TCAM 实现模式匹配需使用价格昂贵的专用的 CAM 存储器，不适合推广应用。

通过对模式匹配算法中 Hash 算法的研究，针对 FPGA 并行与流水线的特点，本文采用

HashMem 算法实现模式匹配。此基础上,本文设计并实现了一种基于 Snort 规则,应用 FPGA 技术,以 HashMem 算法为核心的模式匹配系统。针对 Hash 算法中出现的冲突问题,文中引进了预处理的思想。通过预处理,找出冲突的模式串对其进行单独的匹配从而解决了冲突。

最后,本文对模式匹配系统进行了测试。测试从数据吞吐量、消耗硬件资源两个方向进行。测试结果表明系统具有较高的处理速度,较大的数据吞吐量,更为重要的当规则数不断增加时,系统对硬件资源的消耗增加很少。

2 基于规则的模式匹配系统

2.1 HashMem 算法

HashMem 算法是通过基于异或操作(\oplus)的 Hash 函数 $h=f(x)$ 将集合 X 中的每个元素映射到另一个随机空间 H 中,函数值 h 为相应元素 x 的索引值。通过索引值 h 一次查找即可获得数据 x 。

基于 FPGA 的 HashMem 算法结构如图 1 所示。设定一个输入缓冲,该缓冲为一个基于字节的移位窗口,每一个时钟周期向前移动一个字节^[6]。该窗口的长度为 2 倍字符串长度。输入的字符串经过 Hash 计算模块得出一个地址,用该地址从模式串存储器(所有模式串都按照其对应的 Hash 地址存放到存储器中,称该存储器为模式串存储器)读取数据,送入比较器;这时参与 Hash 计算的字符串正好移动到缓冲的下半部分,进入到比较器中,经过比较,产生匹配信息。这样的设计正好使系统变成流水线结构,保证每一个时钟周期能够输入到系统一个字符串。

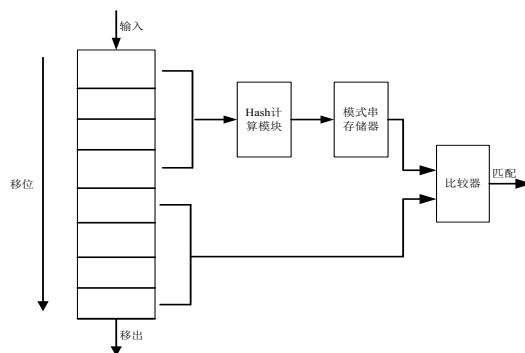


图 1 HashMem 匹配结构

Hash 函数的选择是非常关键的一个环节。FPGA 适合于处理简单的逻辑运算,所以选择的 Hash 函数应由简单的逻辑运算组成^[5]。

对于一个有 n 个比特的位串 X ,可以用式(1)表示。

$$X = \langle x_0, x_1, x_2, x_3, \dots, x_{n-1} \rangle \quad (1)$$

对 X 产生的 Hash 函数如式(2)所示:

$$h(X) = d_0 x_0 \oplus d_1 x_1 \oplus d_2 x_2 \oplus \dots \oplus d_{n-1} x_{n-1} \quad (2)$$

式中,“ \cdot ”为与门,“ \oplus ”是异或门, d_i 表示一个基于随机公式的随机数阵列。由于在构造 Hash 函数时,式(2)只使用了非常适合 FPGA 实现的与门和异或门,因此该 Hash 函数可以用于 HashMem 匹配算法中。

HashMem 函数中的随机数通过式(3)(4)获得。

$$b_i = (\alpha \cdot b_{i-1} + c) \bmod M \quad (3)$$

$$\zeta_i = b_i / M, \quad i = 1, 2, 3, \dots, n \quad (4)$$

式中 α 是小于 M 的正整数, c 是非负的整数, ζ_i 是要获得的随机数。通过统计得出,当 $M = 2^{31}-1$, $\alpha = 16087$ 或 630360016 时,获得的随机数 ζ_i 具有较好的随机性。初始值 b_0 通过 Visual c++ 中的随机数函数 `srand()` 和 `rand()` 产生。通过式(3)(4)能获得一系列取值范围在 0 到 1 之间的随机的有理数 ζ_i 。

2.2 Snort 规则

Snort 是一个轻量级入侵检测系统。当网络包中某数据与其相匹配时，则此数据为入侵数据。Snort 中的一个规则如下：

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 8888 (msg:"P2P napster login"; flow: to_server, established; content:"ec|00 02 00"; offset:1; depth:3; classtype:policy-violation; sid:549; rev:6;)
```

该规则是识别所有从内部网络到外部网络 8888 端口中包含有字符串“ec|00 02 00”的 TCP 数据包，然后报警^[7]。其中“ec”用 ASCII 表示，|00 02 00|用 16 进制数表示。

Snort 的更新速度很快，从 2003 年到 2008 年 Snort 共发布了 8 个版本的入侵检测规则，模式串从 1,942 增长到了 3,563 个。

2.3 系统方案确定

基于以上分析，Snort 更新频繁，基于 FPGA 的系统具有处理速度快，易于调整，能够根据需要重新配置硬件功能的特点，因此用 FPGA 实现系统。HashMem 算法适合于 FPGA 实现，文中用 HashMem 算法实现模式串匹配。

若 FPGA 中匹配的模式串太长，如实现 100 字节长度的模式串匹配，每次需要进行 100 次 Hash 操作，消耗太多的计算资源。Snort 2.0 中模式串长度大于 16 字节的模式串只占总数的 16.6%，且入侵数据在网络数据中的比例很小，我们用 FPGA 实现模式串长度为 1 到 16 字节的模式匹配，大于 16 字节的用软件实现。

3 基于 FPGA 的模式匹配系统实现

本系统是基于 Altera 公司的 Cyclone II EP2C70F896C6N FPGA 芯片，其含有 68,416 个逻辑单元，能生成片上双端口 RAM。由于 SOPC 中提供各种 IP，具备软硬件在系统可编程的功能，所以我们采用 SOPC 技术实现系统。

3.1 系统架构

硬件系统由 Nios II 处理器、以太网控制器 DM9000A、存储器与定制 IP 核组成，如图 2 所示。其中定制 IP 用于模式串匹配。双口 RAM 用于存储接收到的网络数据，Nios II 处理器与定制 IP 均与之相连，在 Nios II 接收网络数据的同时，定制 IP 自动读取双口 RAM 中存储的待处理数据进行匹配，达到并行效果。模式串根据相应 Hash 地址存入 SDRAM。当非法数据检测出后，通过以太网传给上位机，由上位机进一步对其处理。

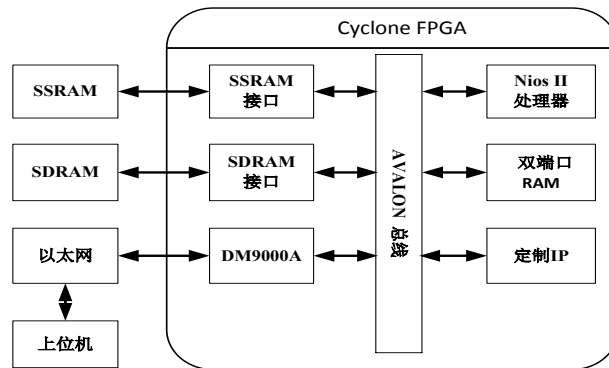


图 2 入侵检测硬件系统结构

SOPC Builder 中的系统连接图如图 3 所示。

Use	C...	Module Name	Description	Clock	Base	End	Tags
✓		cpu_0	Nios II Processor	clk	0x01915000	0x019157ff	
✓		tri_state_bridge_0	Avalon-MM Tristate Bridge	clk			
✓		onchip_mem	On-Chip Memory (RAM or ROM)	clk	0x01908000	0x0190c7ff	
✓		dma_0	DMA Controller	clk	0x01916860	0x0191687f	
✓		check_top0	check_top	clk	0x01916000	0x019167ff	
✓		onchip_send	On-Chip Memory (RAM or ROM)	clk	0x01912000	0x01913bff	
✓		cfi_flash_0	Flash Memory Interface (CFI)	clk	0x01400000	0x017fffff	
✓		sdram	SDRAM Controller	clk_50	0x00800000	0x00ffffff	
✓		epcs_controller	EPCS Serial Flash Controller	clk	0x01915800	0x01915fff	
✓		jtag_uart_0	JTAG UART	clk	0x01916a00	0x01916a07	
✓		uart_0	UART (RS-232 Serial Port)	clk	0x01916800	0x0191681f	
✓		timer	Interval Timer	clk	0x01916820	0x0191683f	
✓		timer_1	Interval Timer	clk	0x01916840	0x0191685f	
✓		lcd_16207_0	Character LCD	clk	0x01916880	0x0191688f	
✓		led_red	PIO (Parallel I/O)	clk	0x01916890	0x0191689f	timer_1
✓		led_green	PIO (Parallel I/O)	clk	0x019168a0	0x019168af	Interval Timer [altera avalon time
✓		button_pio	PIO (Parallel I/O)	clk	0x019168b0	0x019168bf	
✓		switch_pio	PIO (Parallel I/O)	clk	0x019168c0	0x019168cf	
✓		sram	SRAM_16Bit_512K	clk	0x01880000	0x018fffff	
✓		DM9000A	DM9000A	clk	0x01916a08	0x01916a0f	
✓		SD_DAT	PIO (Parallel I/O)	clk	0x019168d0	0x019168df	
✓		SD_CMD	PIO (Parallel I/O)	clk	0x019168e0	0x019168ef	
✓		SD_CLK	PIO (Parallel I/O)	clk	0x019168f0	0x019168ff	
✓		cpu_arp_data	PIO (Parallel I/O)	clk	0x01916900	0x0191690f	
✓		cpu_arp_done	PIO (Parallel I/O)	clk	0x01916910	0x0191691f	

图 3 系统 SOPC 连接图

Nios II 中软件实现并行效果，Nios II 中网络数据的接收与定制 IP 核的处理并行执行，流程如图 4 所示。

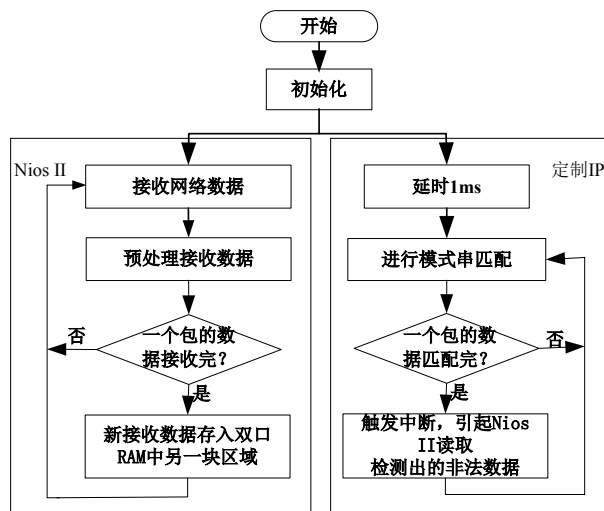


图 4 并行处理流程图

Nios II 中开始时进行初始化，计算 Snort 中规则的 Hash 地址，将其映射存储入 SDRAM（本系统用 SDRAM 作为 Hash 散列表）。然后启动网络接口接收数据，对接收的数据进行预处理，存入双口 RAM，若接收完一个数据包，则新接收的数据存入双口 RAM 另一块区域中(双口 RAM 中开辟了俩块存储区域)；与此同时并行进行定制 IP 的运行，初始启动时延时 1ms (延时 1ms 使网络接口接收到一定量的数据)，然后读取双口 RAM 中的待处理数据进行 HashMem 匹配，若处理完一个数据包，则触发中断，引起 Nios II 读取检测出的入侵数据，通过网络接口将匹配信息发送给上位机进行进一步处理，然后定制 IP 读取双口 RAM 中另一块区域数据。如此不断进行。

3.2 以太网收发模块

系统采用以太网控制器 DM9000A 实现 FPGA 的以太网接入。在 SOPC Builder 中设计用户接口电路与 DM9000A 连接，在 Nios II 中编写以太网接口驱动程序，即可实现以太网通讯。以太网一个完整的驱动程序包括：以太网控制芯片的初始化，数据包的发送，数据包的接收。

3.2.1 芯片的初始化

DM9000A 正常工作需要在上电后对内部寄存器进行初始化，该过程通过 FPGA 对 DM9000A 外部控制总线 and 数据总线的读写操作完成。具体流程如下所示：

- (1)激活内部 PHY,上电复位后系统默认是关闭内部 PHY,通过写 0 到 0x1F 寄存器激活 PHY。
- (2)软复位 DM9000A 初始化,通过写 1 到 0x00 寄存器,延时 10 μ s 对其进行清零。
- (3)配置 NCR 寄存器(NCR (REG_00) LBK Bit[2:1] = "00"b)使其工作在正常模式。
- (4)配置 IMR 寄存器(REG_FFH Bit [7]=1),启用读写缓存地址指针自动返回功能。
- (5)配置物理地址寄存器(REG_10H~REG_15H),写入 6 Byte 以太网 MAC 地址到物理地址寄存器。
- (6)配置 IMR 寄存器,开启接收、发送中断。
- (7)配置 RCR 寄存器使能接收。
- (8)DM9000A 的 NIC 被激活,等待接收发送数据。

3.2.2 数据包的发送

在 DM9000A 中有 3KB 的静态 RAM 作为发送缓冲区，其中可同时保存 2 个完整的以太网帧。以下是发送一个数据包的具体步骤：首先，利用写操作寄存器 MWCMD 向 DM9000A 的发送缓存区中写入发送数据帧；其次，把数据帧的长度写入寄存器 FCH 和 FDH；然后，发送数据发送请求，设置 TXREQ 为“1”；最后，检查 TSR，判断此帧数据是否发送完。数据包发送流程图如图 5 所示。

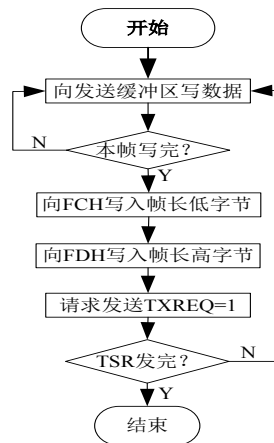


图 5 数据包发送流程图

3.2.3 数据包的接收

接收到的数据先保存在 DM9000A 内部 SRAM 中地址从 0x0C00~0x3FFF 的 13 KB 空间中。若是读取位置超过 3FFFh 时，DM9000A 会自动将位置移到 0C00h 的位置。在每一个数据包前，会增加 4 字节存放数据包相关资料。通过读这 4 个字节确定数据包的状态^[8]。数据包接收的流程如图 6 所示。

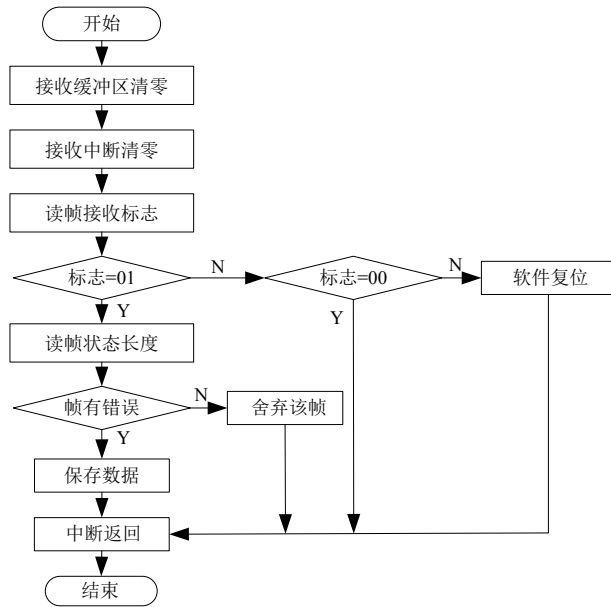


图 6 数据包接收流程图

3.3 模式匹配模块

入侵检测系统的核心部分是模式串匹配模块。本文采用 HashMem 算法，以定制主外设实现模式串匹配。

定制 IP 应按照 Avalon 总线规范设计，符合 Avalon 总线规范的逻辑才能正常工作于 SOPC 系统中。Avalon 总线主要用于连接片内处理器与外设，以构成片上可编程系统（SOPC）。它描述了主从构件间的端口连接关系，以及构件间通讯的时序关系。Avalon 总线规范提供了各种选项，来剪裁总线信号和时序，以满足不同类型外设的需要。

3.3.1 模式匹配 IP 核架构

Avalon 外设分为主外设和从外设两类。由于模式匹配模块需在处理过程中不断读取存储器中的数据，所以设计为主外设。为了使 Nios II 及时响应 IP 的请求，本 IP 采用了中断机制。

模式匹配的定制 IP 架构如图 7 所示。当定制 IP 开启后，读取双口 RAM 中的待匹配数据，计算 Hash 地址，取出 SDRAM 对应 Hash 地址中模式串与当前匹配数据进行匹配，得出匹配信息，并记录入侵数据在网络包中的位置。当匹配完一个数据包，则触发中断，引起 Nios II 读取匹配信息。

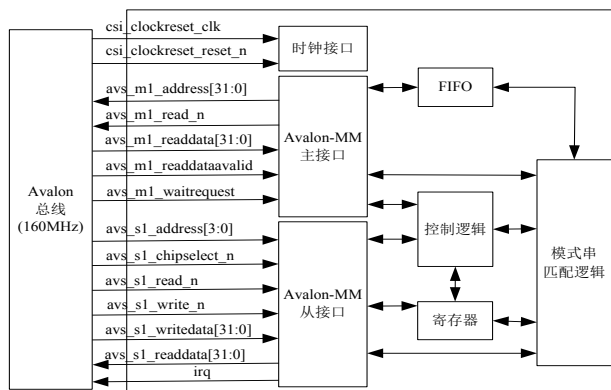


图 7 模式串匹配 IP 架构

IP 核的端口信号如图 8 所示。

Name	Interface	Signal Type	Width	Direction
clk	clock_reset	clk	1	input
reset	clock_reset	reset	1	input
slave_address	slave	address	3	input
slave_read	slave	read	1	input
slave_readdata	slave	readdata	32	output
slave_write	slave	write	1	input
slave_writedata	slave	writedata	32	input
slave_byteenable	slave	byteenable	4	input
master_address	avalon_master	address	32	output
master_read	avalon_master	read	1	output
master_readdata	avalon_master	readdata	32	input
master_byteena...	avalon_master	byteenable	4	output
master_readdata...	avalon_master	readdatavalid	1	input
master_waitrequ...	avalon_master	waitrequest	1	input
irq	interrupt_sender	irq	1	output

图 8 SOPC Builder中IP核的端口信号

本文用上位机处理大于 16 字节的模式串匹配。当上位机接收到 16 字节的入侵数据，则根据其位置信息，对此 16 字节及紧随其后的数据进行 HashMem 算法匹配。因为 16 字节以上的入侵数据很少，所以上位机的处理速度高于 FPGA 速度，系统速度由 FPGA 决定。

3.3.2 HashMem 算法实现

在系统架构中，模式串匹配逻辑实现 HashMem 算法匹配功能。其内部结构示意图如图 9 所示。

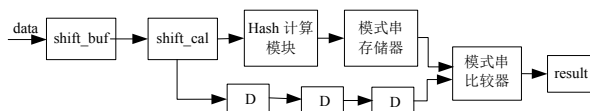


图9 HashMem算法匹配结构示意图

移位寄存器 shift_cal 为 16 字节，存储当前匹配处理字符串，其数据能组成 1 到 16 字节共 16 个字符串(每次均需匹配 shift_cal 中 16 字节所组成的 1 到 16 字节长的 16 个字符串)。移位寄存器 shift_buf 中为从双口 RAM 中新读取的待处理数据，其流水式移位更新寄存器 shift_cal。shift_cal 中数据经过 Hash 计算模块得出 Hash 地址，然后从模式串存储器相应 Hash 地址中取出模式串。shift_cal 中数据经过三个时钟与刚从模式串存储器中取出的模式串进行比较，得出匹配信息，存储于寄存器 result 中。

3.3.3 HashMem 冲突的解决

采用 HashMem 算法时会出现 Hash 冲突，给不同长度的模式串分配不同的 Hash 函数与不同的存储区域，对个数较多的长度的模式串分配更多的存储区域，使冲突降到最低。

为了进一步解决 Hash 冲突，在 PC 上对 Snort 规则进行预处理，步骤如下：

- (1)从 Snort 规则中提取模式串；
- (2)生成基于随机数公式(3,4)的随机数；
- (3)根据模式串长度计算出基于式 (1,2) 的 HashMem 函数的 Hash 值；

(4)找出冲突的模式串进行保存;

(5)反复进行步骤(2)到(4),选取冲突最少的 HashMem 函数,输出整理后除去冲突的 Snort 规则。

经过预处理后,本系统只有 4 字节、14 字节、16 字节长度模式匹配时存在冲突,且每个长度冲突只有一个,分别为“Girl”、“Content-Type\.”、“PGV_BASE_DIRECTORY”。由于冲突很少,所以对冲突的数据再进行单独的匹配,即使 4 字节、14 字节、16 字节三个长度的当前匹配数据在参加 HashMem 算法匹配同时,又与冲突数据“Girl”、“Content-Type\.”、“PGV_BASE_DIRECTORY”进行匹配,若匹配,则直接判定为非法数据;若不匹配,则采用 HashMem 算法匹配信息。如此解决了冲突。

解决了冲突的模式串匹配 IP 的示意图如图 10 所示。

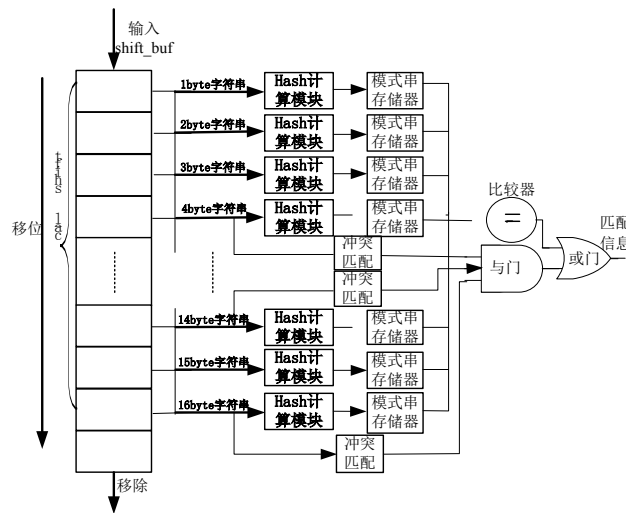


图 10 解决冲突的模式串匹配 IP 示意图

3.3.4 模式串匹配 IP 匹配过程

若匹配模式串“abcdefghijklmnoqrst”，则开始寄存器 shift_cal 内存储“abcdefghijklmno”，shift_buf 内存储“qrst”（图 11），然后对“a”、“ab”、“abc”、“abcd”直到“abcdefghijklmno”共 16 个模式串同时并行进行 HashMem 算法匹配，计算 Hash 地址，从 Hash 散列表相应地址中取出模式串进行匹配（图 12 中 T1 即为 Hash 散列表，每个地址存储 4 字节数据），得出匹配信息（图 12）。

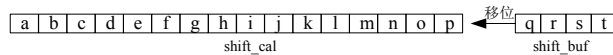


图 11 第一轮匹配时 shift_cal、shift_buf 中内容

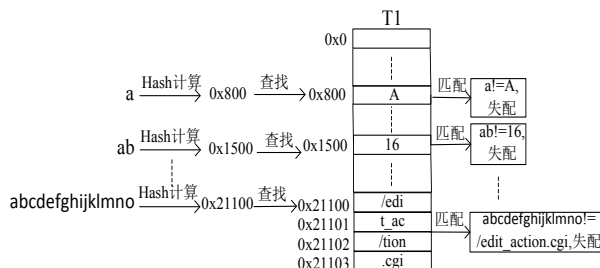


图 12 HashMem 算法匹配

与此同时,4 字节、14 字节、16 字节长度的字符串“abcd”、“abcdefghijklm”、“abcdefghijkl

lmno”分别与冲突模式串“Girl”、“Content-Type \.”、“PGV_BASE_DIRECTORY”进行匹配，此次不匹配，结果采用 HashMem 算法匹配信息（图 13）。接着 shift_buf 更新 shift_cal 为“bcdefghijklmnopq”，进行新一轮匹配（图 14）。如此直到 shift_cal 为“efghijklmnopqrst”得到全部匹配信息。

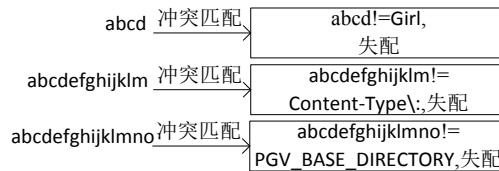


图 13 冲突处理

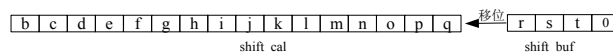


图 14 第二轮匹配时 shift_cal、shift_buf 中内容

3.4 系统运行过程

3.4.1 以太网测试文件发送程序

在 VC++6.0 中编写上位机程序控制以太网发送测试数据。在发送测试数据网络包中，我们在包头部加入特殊字符：FPGA 开发板的 MAC 地址{0x01,0x60,0x6E,0x11,0x02,0x0F}，以区别普通的以太网帧。在测试文件中，每次都会加入一些 Snort 规则，测试系统能否全部正确检测出。发送程序界面如图 15 所示。

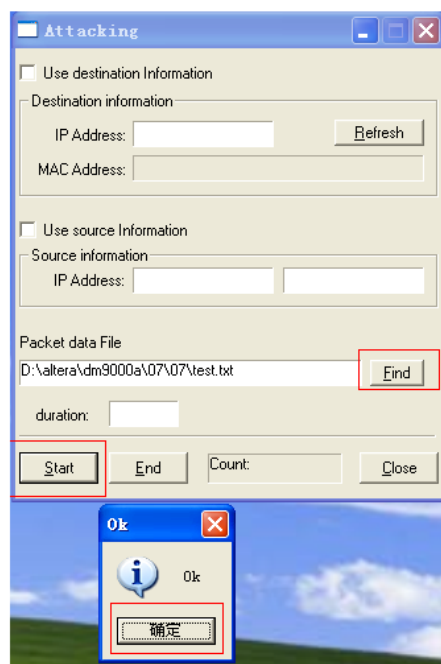


图 15 以太网测试程序发送界面

3.4.2 上位机结果处理程序

结果显示程序是用 VC 图形化窗口实现的，程序运行环境是 VC++6.0，主要功能是：

- (1)打开串口，接收由 FPGA 发送过来的串口数据。
- (2)对 16 字节以上的数据进行再次匹配，判断其是否为入侵数据。
- (3)获取源文件并根据所接收的数据，以不同颜色显示 FPGA 检测到的非法数据。

程序界面如图 16 所示。

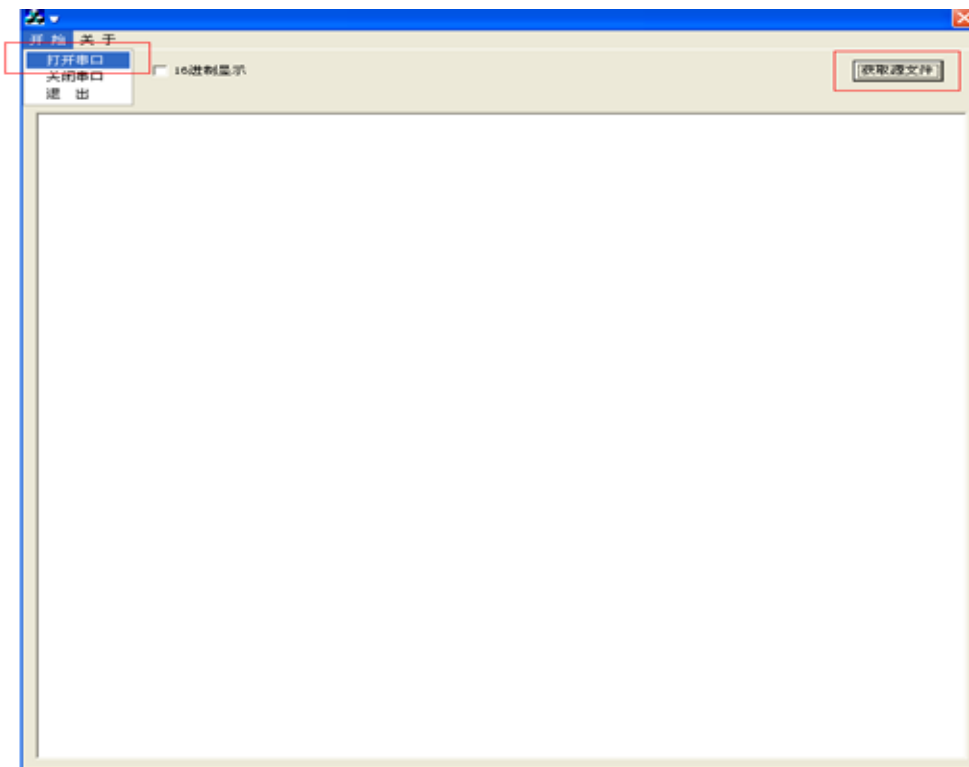


图 16 结果显示程序界面

程序在对检测结果显示时，为了加强显示效果运用了红绿蓝三种颜色交替显示。

程序运行结果如图 17 所示。以不同颜色显示了非法字符串，并且在原测试文件的查找
到非法字符。

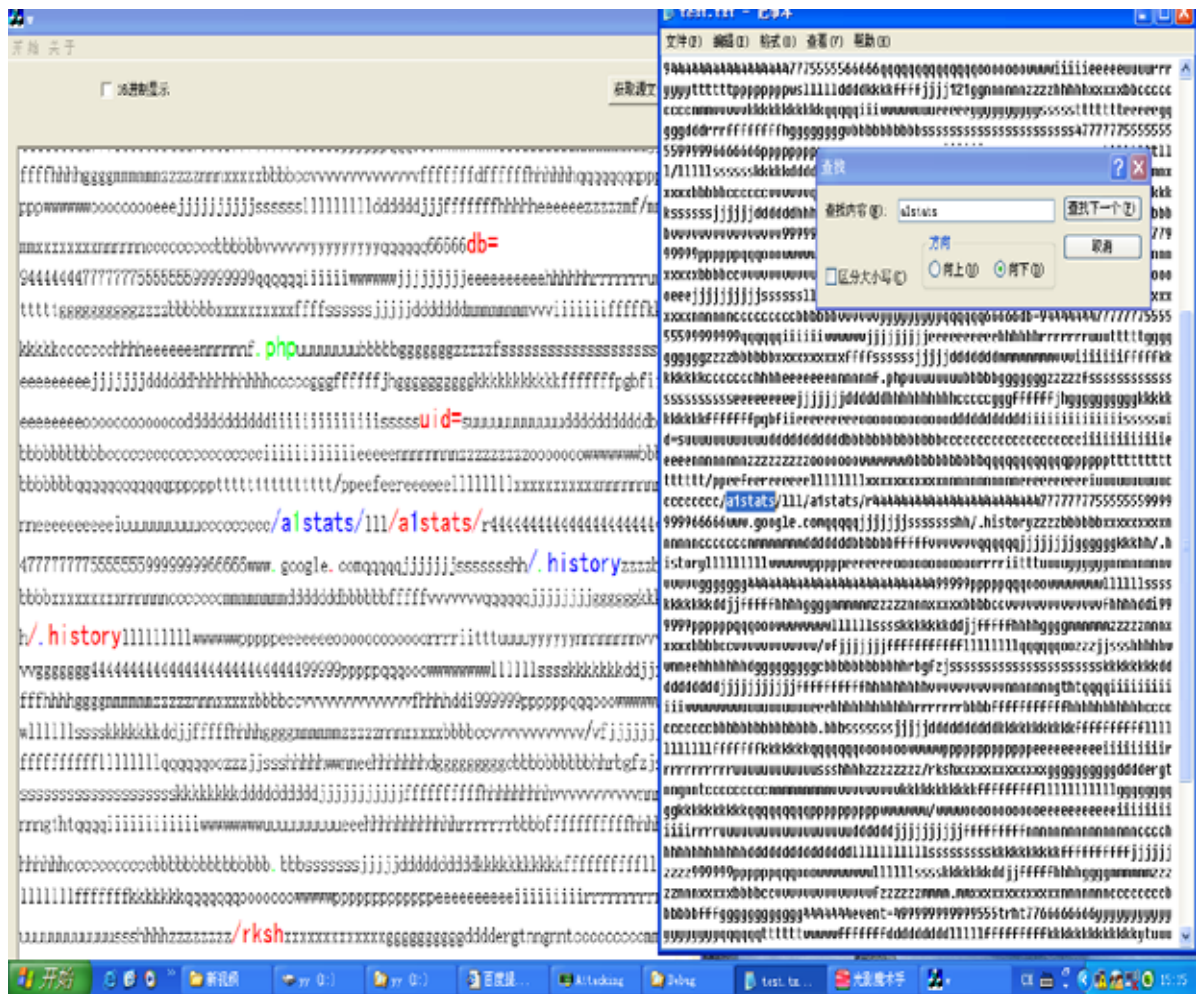


图 17 程序运行结果

4 系统性能

4.1 系统时序结果

系统时钟为 100MHz，Quartus 中时序分析结果如图 18 所示。

Timing Analyzer Summary					
	Type	Slack	Required Time	Actual Time	From
1	Worst-case tsu	1.659 ns	5.000 ns	3.341 ns	SRAM_DQ[8]
2	Worst-case tco	-4.570 ns	5.000 ns	9.570 ns	nios_iu2lsram
3	Worst-case tpd	2.388 ns	5.000 ns	2.612 ns	altera_internal
4	Worst-case th	N/A	None	3.064 ns	altera_internal
5	Clock Setup: 'SDRAM_PLL:u1altpll:altpll_compo...	0.923 ns	100.00 MHz (period = 10.000 ns)	110.17 MHz (period = 9.077 ns)	nios_iu2lcpu
6	Clock Setup: 'CLOCK_50'	0.971 ns	50.00 MHz (period = 20.000 ns)	N/A	nios_iu2lnios
7	Clock Setup: 'altera_internal_itag~TCKUTAP'	N/A	None	131.13 MHz (period = 7.626 ns)	sld_hub:auto_l
8	Clock Hold: 'SDRAM_PLL:u1altpll:altpll_compo...	0.391 ns	100.00 MHz (period = 10.000 ns)	N/A	nios_iu2lchec
9	Clock Hold: 'CLOCK_50'	0.391 ns	50.00 MHz (period = 20.000 ns)	N/A	nios_iu2lsdrar
10	Total number of failed paths				

图 18 系统时序分析结果

4.2 系统测试结果

4.2.1 HashMem 算法软、硬件吞吐量测试

在 PC 机和 FPGA 芯片上分别进行基于 Snort 2.0 入侵检测规则的 HashMem 算法的实现，

对吞吐量进行测试, 结果如表 1 所示。

PC 测试环境: Intel(R) Pentium (R)4 CPU 主频 1.70 GHz, 内存 512MB, 操作系统为 Microsoft Windows XP Professional

系统: Altera 的 Cyclone II EP2C70F896C6N FPGA 芯片, 其含有 68,416 个逻辑单元, 支持双端口 RAM, 主频 100Mhz, 2M SSRAM, 32*2M SDRAM。

由表 1 可知, 基于 FPGA 系统的吞吐量为 61Mbps, 大于软件方法的 35Mbps。

表 1 HashMem 算法软、硬件吞吐量测试(Mbps)

规则数 \ 实现方式	800	1200	1600	2000
软件	36	35	35	35
FPGA	61	61	61	61

4.2.2 系统硬件测试结果

系统硬件测试详细结果如表 2 所示。

表 2 基于 FPGA 的多种模式匹配算法对比测试结果

算法	目标器件	最高频率 (MHz)	实现字符数量	使用逻辑资源	吞吐量 (Mbps)
HashMem	EP2C70F896C6N	130	15,311 30,927 60,930	7,822 8,410 8,998 (LE)	61

由表 2 可知, 系统采用的 HashMem 算法的匹配与模式串数量相关不大, 系统延时未随字符串增加而增加, 当匹配字符数量从 15,311 增加到 60,930 时, 系统吞吐量为 61Mbps、最高频率为 130MHz 不变, 使用逻辑单元从 7,822 LEs 增到 8,998 LEs, 增加很少。当字符数量更多时, 本文与传统系统随着字符串增加性能下降相比更加优越。

5 结论

针对目前入侵检测系统中, 模式匹配算法软件实现方式和专用硬件电路实现的处理速度, 资源占用, 规则更新方面的不足, 本文设计实现了一种基于 FPGA 的采用 HashMem 算法, 使用 Snort 规则的网络入侵检测中模式匹配系统, 吞吐量为软件方法的 2 倍以上。当规则数增加时, 如从 400 增加到 2000, 吞吐量为 61Mbps、最高频率为 130MHz 不变, 逻辑单元增加很少, 与传统系统随着规则数增加性能下降相比更具优势。

本文的特点主要为在 FPGA 上实现了 HashMem 算法; 用硬件方法解决了冲突, 不会造成误判; 采用由异或和与运算组成的 Hash 函数, 适合于 FPGA 快速处理; 用上位机处理 16 字节以上的模式串匹配, 使 FPGA 处理速度有较大提高。

参考文献

- [1] Janardhan Singaraju, John A. ,Chandy. FPGA based string matching for network processing application [J]. Microprocessors and Microsystems, 2008, 32(4): 210-222.
- [2] Katashita, T, Maeda, A, Toda, K, , et al. A Method of Generating Highly Efficient String Matching Circuit for Intrusion Detection [A], Field Programmable Logic and Applications[C]. Tsukuba, Japan, 2006: 1-4.
- [3] Bispo J, Sourdis I, Cardoso J M P, et al. Regular expression matching for reconfigurable packet inspection [A], Field Programmable Technology[C]. FPT 2006. IEEE International Conference on, 2006:119 - 126.
- [4] I Sourdis, D Pnevmatikatos. Pre-decoded cams for efficient and high-speed NIDS pattern matching [A], the 12th IEEE Symposium on Field-Programmable Custom Computing Machines[C], Chania, Greece, 2004:258-267.
- [5] Tran Ngoc Thinh, Kittitornkun S, Tomiyama S. Applying Cuckoo Hashing for FPGA-based Pattern Matching in NIDS/NIPS[A], Field-Programmable Technology [C]. ICFTPT. International Conference on,

2007:121 - 128.

- [6] 黄建. 入侵检测系统中字符串匹配算法与实现[D]. 湖北武汉:华中科技大学,2008,3.
Huang Jian. Algorithms and Implementation of String Match in Intrusion Detection System [D] . Wuhan, Hubei: Huazhong University of Science and Technology of China,2008,3. (in Chinese)
- [7] United Microelectronics Corporation. DM9000A Programming Guider[K/OL]. (2006-10-30) [2010-03-31].
<http://www.davicom.com.tw>, 2006-10-30.
- [8] 陈小毛,陈尚松.32 为软核处理器 NIOS II 的以太网接口设计与实现[J].电子测量技术,2007,30(1):150-151,187.
- [9] 张克农,陆佳华,常羽飞. 入侵检测系统中高速字符串匹配协处理的实现方法[J].微电子学与计算机,2006, 23(4):35-38.
- [10] Christopher R Clark, David E. Schimmel. A Pattern Matching CO-Processor for Network Intrusion Detection Systems [A] , 2003 IEEE International Conference on Field-Programmable Technology [C]. Atlanta, GA, USA, 2003:68 -74

原创性声明：本论文《基于 SOPC 的网络入侵检测中模式匹配系统设计》是作者在东北大学攻读硕士研究生期间，在导师的指导下进行研究工作所取得的成果。除文中已注明的引用内容外，不包括其他人已经发表或撰写过的研究成果。

作者签名:陈勇 任鹤翔 南巧玲

导师签名:李晶皎

撰写日期:二零一一年九月二十九日

补充：本文系统与作者参加的 Altera 2011 亚洲创新大赛中的作品类似。