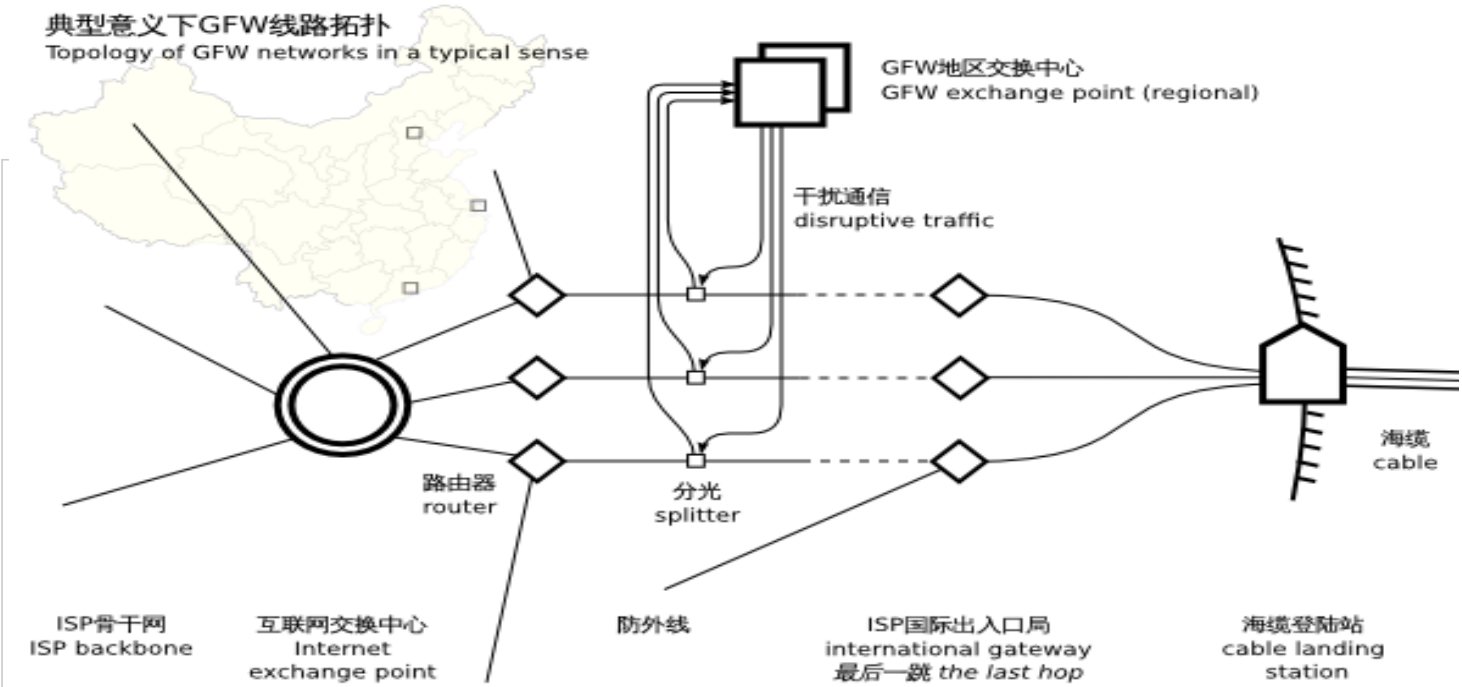
深入理解GFW：内部结构

之前我们对GFW进行了大量的黑箱测试，尽管大多数实验数据都得到了良好的解释，但是还是有些数据或者体现出的规律性（不规律性）没有得到合理的解释。比如TCP连接的各项超时时间，比如Google的443端口被无状态阻断时，继发状态的持续跟源IP相关的问题。比如一般TCP连接的继发阻断时，窗口尺寸和TTL的连续变化特性。这些问题已经超出纯协议的范畴，需要对GFW的内部结构进行进一步了解才能明白其原因。所以在这一章介绍GFW的实现和内部结构。

总的来说，GFW是一个建立在高性能计算集群上规模庞大的分布式入侵检测系统。其分布式架构带来了很高的可伸缩性，对骨干网一点上庞大流量的处理问题被成功转换成购买超级计算机堆砌处理能力的问题。它目前有能力对中国大陆全部国际网络流量进行复杂和深度的检测，而且处理能力“还有很大潜力”（2009年8月）[null]。

线路接入



svg

对于GFW在网络上的位置，有很模糊的认知：“在三个国际出口作旁路监听”。然而还希望对在出国之前最后一跳之前发生了什么有详细了解。GFW希望对不同线路的链路异构性进行耦合，并研究了快速以太网、低速WAN、光纤、专用信号多种类型链路的耦合技术。[o3b]而根据《国际通信出入口局管理办法》，几大ISP有自己的国际出入口局，最后在公用国际光缆处汇合，比如在海缆登陆站之前汇合。据已有的资料，安管中心（CNNISC）有独立的交换中心，而且有报道说各个ISP是分别接入其交换中心。这样几个材料就可以形成一致的解释：为了适应不同ISP不同的链路规格，GFW自己的交换中心需要对不同的链路进行整合，不同的ISP分别引出旁路接入GFW。而没有接入GFW的线路则被称为“防外线”[来源不可靠]，不受GFW影响。接入的线路类型应该主要是光纤线路，因此通常称此接入方式为分光。这就是“旁路分光”。另外实验发现，GFW的接入地点并不一定紧靠最后一跳，因此图中以虚线表示。需要注意GFW的响应流量重新接回网络的地点难以确认，这里只是假设是与接出的地点相同。

负载平衡

面对多条骨干监测线路接入产生的巨大不均匀流量，不能直接接到处理集群，而是要先进行汇聚然后再负载均衡分流成均匀的小流量，分别送给处理集群并行处理。首先需要将网络设备通信接口（Pos、ATM、

E1等）转换成节点可用的主机通信接口（FE、GE等）。处理负载均衡的算法经过仔细考虑，希望实现：流量均匀分布、对于有连接协议保持连接约束、算法简单。连接约束是指：一对地址端口对之间的一个连

接全部通信都要保证调度到同一个节点。[03b][03a][05a]

GFW关于负载平衡的文章中主要提出两种算法。一种是轮转调度，对于TCP，当SYN到达时，以最近分配

的节点号取模再加1，并将连接存入hash表，当后继流量到达时就能查询hash表获得目标节点号。[03a]另一种是基于连接参数的散列，对于N个输出端口调

度输出端口号是H(源地址，目标地址，源端口，目标端

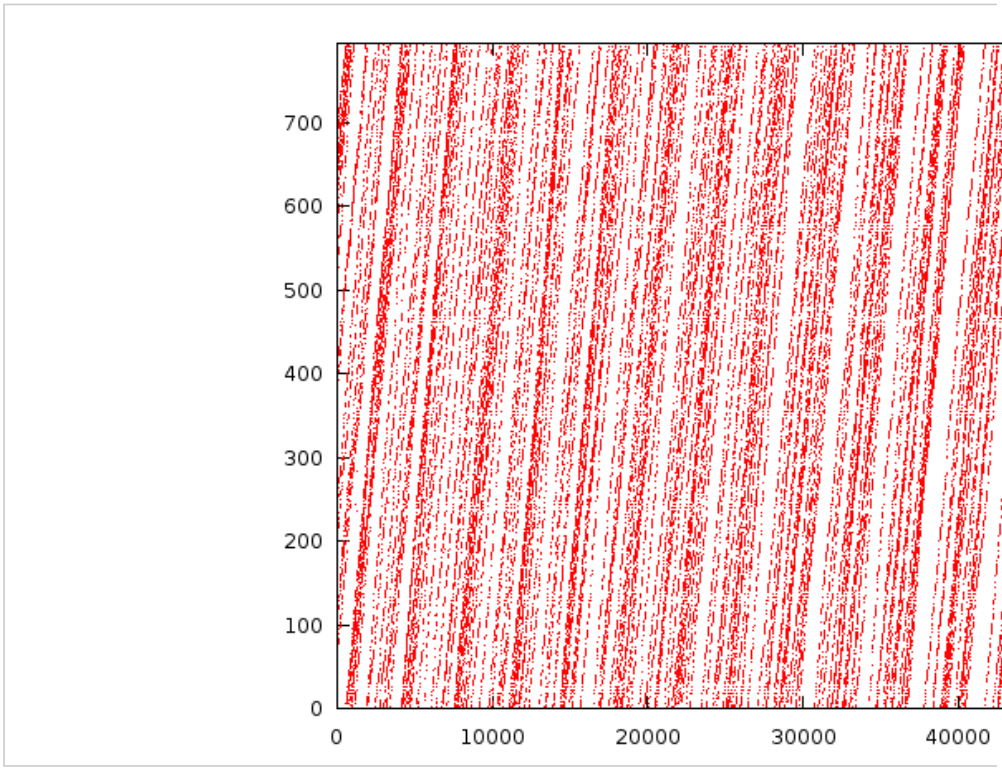
口) mod N[03b]，这个H函数可以是xor[05a]。

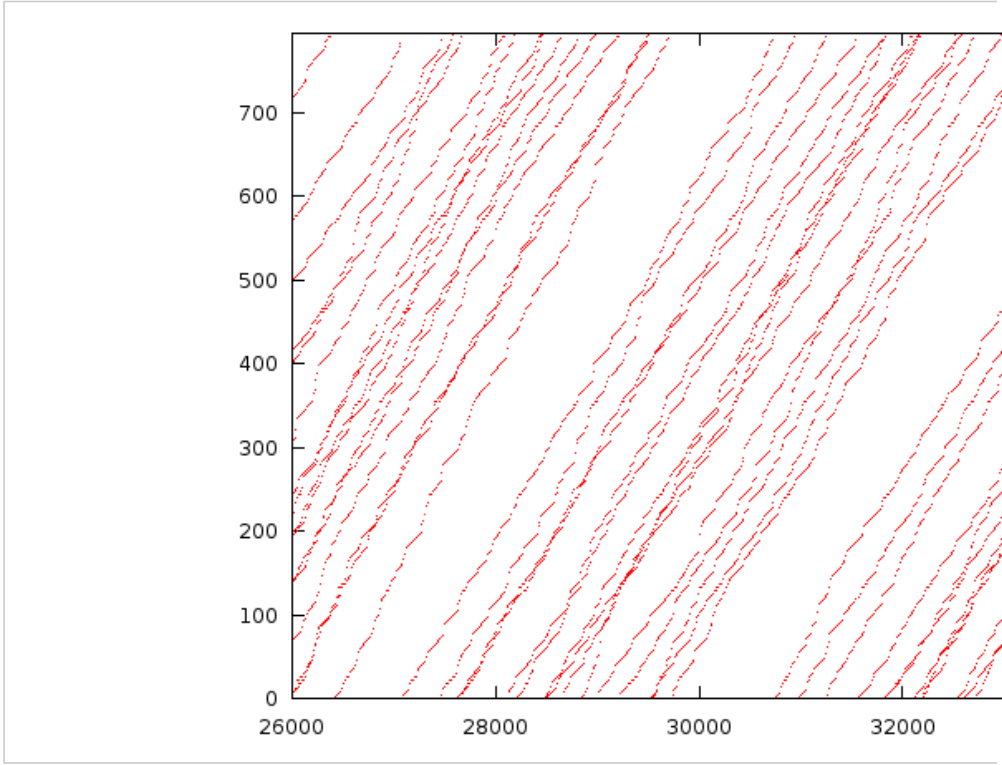
而之前的某个实验中我们碰到一种特殊的模式，负载平衡在解释其现象中起到了重要作用，下面专门分出一节详细说明。

一个关于窗口值的实验

实验步骤：发送含有关键词的特制包通过GFW，并接收GFW返回的阻断响应包。因为触发阻断之后，同地址对和同目标端口的连接都会受到继发阻断，为了消除这种干扰，一般采取顺序改变目标端口的扫描式方法。通过前期一些实验，我们已经发现和确认某类（二型）阻断响应包中的TTL和id都跟窗口大小有线性关系，我们认为窗口是基本量（二型窗口为5042时id发生了溢出，只有在id根据窗口算出时才会发生此种情况）。

然而在顺序扫描中有一种特殊的模式无法用现有证据解释。进一步的实验步骤是：在源、目标地址不变的情况下，顺序扫描目标端口，记录返回的阻断响应包的窗口。数据如下图，横轴是时间（秒），纵轴是端口号，每个点代表一次阻断触发事件中观测者收到的阻断包的窗口值。





可以看出，在同一时间有13根较连续的线。这样产生了几个问题：为什么有独立可区分的不同的线？这些线表示了什么？为什么有13根？为什么每根线是递增的？

为什么有独立可区分的不同的线？现象具有明显的可以继续划分的子模式，而不是一个整体的随机量，并且每个子模式都有良好的连续增加的性质。因此可以推测产生此现象的内在机制不是一块铁板，而是多个独立的实体。进一步的实验事实是，如果顺序扫描端口每次增加13，那么只会产生一条较连续的线而排除其他的线。这直接证明了模13同余端口产生结果的不可分性、实体性，以及同余类间的独立性。
这些线表示了什么？我们猜想，这13根线就表征了背后有13个独立实体分别根据某个内在的状态产生阻断响应，窗口值就是其内在状态的直接表现。
为什么有13个？而不是1个2个？这个时候，负载平衡就是对此事实的一种解释良好的模型。如果GFW有13个节点在线，由于希望将流量平均分配到每个节点，那么根据前面论文所述，便采用模的方式，在源、目标地址不变时，根据目标端口模13分配流量，目标端口模13同余的包会进入同一个节点。实际上更早的时候的一次实验是发现有15根线，同理可以猜测有15个节点在线。
为什么每根线是递增的？实验中发现，每次阻断GFW会分别向连接双方发送窗口值依次增加的两组阻断包，这样对于每方来说，每次阻断就会使窗口值增加2。每根线会递增正是说明节点在不断产生阻断包增加窗口值，一部分是实验观察者的观测行为触发的，另一部分则是普通网络流量造成的。如果对数据做差分并扣除观测造成的影响，甚至还可以对每节点产生阻断的速率有所估计。但是为什么要让窗口递增？这背后的动机难以找到很合理的解释，可能这个窗口值有计数器的作用，也可能是为了在ip.id上对不同节点产生的包进行区分。事实上，一型的窗口值就是几乎随机但ip.id固定，窗口递增并非是必须的。

然而进一步的实验发现，如果目标端口、源地址不变，而目标地址顺序变化，图像就显得比较紊乱，找不出规律。虽然如此，仍然在局部可以识别出同时存在13根线的情况，进一步证实“13个节点在线”的猜测。这个实验的意义在于，通过对现象的分解约化，分离出GFW内部的某种独立实体结构，对论文中主张的

负载平衡算法有进一步的实践证实，对GFW的内部结构得到进一步的认识。[*]

数据处理

当数据流通过当数据总线到达终端节点之后，需要将其从物理层提取出来供上层进一步分析，这个部分称为报文捕获。普通的做法，先网卡中断一次通知内核来取，然后控制DMA传到内核空间，然后用户用`read()`，让内核`copy_to_user()`将sk_buff的数据复制到用户空间，但是这样复制一次就带来了无谓开销。因此GFW设计环状队列缓存，以半轮询半中断机制减少频繁中断的系统调用开销，用mmap实现

zero-copy，把数据直接从网卡DMA到用户空间。这样性能提高很多（耦合也提高很多）。[**]
链路层数据到怀里了，接下来要将数据上交给TCP/IP栈。论文中多次提到libnids（这个库我们也是第一眼就瞟到了，后来发现对诊断没什么用），将其作为基准，（甚至可能符合国情地）以其为蓝本改进，开发出了一种多线程的TCP/IP（自动机）。后面又在考虑对其进一步做自动机分解优化。后来又再次提出一种两级连接状态记录表，一级轻量级环状hash表可以缓解大量无效连接和SYN Flood的情况，二级表才真正存储连接的信息。实验结果与此相符：发送SYN之后的超时时间要比发送第一个ACK之后的超时时间短得多。文献中还提到libnids的half_stream，从实际的情况上看，GFW的TCP栈的确具有鲜明的半连接特性，也就是说：一个方向的TCP栈只检测客户端到服务端的数据，或者反之。这样一个直接的后果就

是，即使服务端根本不在线没响应，客户端照样可以假装三次握手然后触发一堆RST。往好的方向看，也许是因为多线程TCP栈还原全连接时不想处理数据共享控制的问题。总而言之，GFW有一种非常轻量级的TCP/IP栈，刚好能够处理大多数遵守RFC的连接。如果用户稍微精明一点就能穿过去，GFW要么坐视不管，要么重写TCP栈。[***]

TCP/IP栈将数据分片重组，流重组之后交给应用层解析。应用层由很多插件模块组成，耦合松，部署易。其应用层插件包括“HTTP、TELNET、FTP、SMTP、POP3、FREENET、IMAP、FREEGATE、TRIBOY”。[05a]有意思的是，这是首次官方确认GFW与Freegate、Freenet、Triboy的敌对关系。应用层的协议大家都很熟悉不用多解释，不过应用层问题比传输层更多了。好几个模块都有一些小毛病，比如某类HTTP模块只认得CRLF作为EOL，换作LF便呆了。再比如某类DNS模块，发的DNS干扰包，十有五六都校验和错误，查询AAAA也返回A，还不如关掉。多数模块都是得过且过，刚好可以工作，一点都不完善。这里列出的、发现的问题按照软件设计一般规律也只是冰山一角。由此推断，GFW的设计哲学是：

better is worse。

不过在可以生产论文的话题上，GFW绝不含糊，就是模式匹配。应用层模块把应用层协议解析好了，然后就要看是不是哪里有关键词，字符串匹配。搞了一堆论文出来，改进AC算法和BM算法，就差汇编的干活了，得出某种基于有限状态自动机的多模式匹配算法，特别适合GFW这种预定义关键词的需求。总之复杂度是线性的，攻击匹配算法消耗CPU什么的就不要想了。

响应机制

如果匹配到一个关键词了，要积极响应阻断之。响应的手段其他地方已经说得太多，手懒，特此剽窃一段，欢迎举报学术不正之风：

响应机制的发展已经经历IP包过滤（静态IP包过滤、动态IP包过滤）、连接欺骗（传输层连接欺骗、应用层连接欺骗）两个阶段，并且形成了针对不同的应用多种方式共存的现状。

静态IP包过滤是IDS通过和被保护网络与外部网络之间的连通边的端点网络层设备（路由器、三层交换机等）进行联动，在其上设置访问控制列表（ACL）或静态路由表来实现对指定IP地址的过滤。由于需要过滤的IP地址数量很大，大多数的网络层设备上对ACL大小和性能的支持不能满足要求，因此，实际工作中大多采用静态路由的方式。使用该种方式，信息入侵检测系统只能通过专用客户端程序静态写入的方式进行访问控制，粒度大（IP地址级），响应时间慢，容量较小，但是可以静态写入路由设备的配置文件中，是非易失的。

动态IP包过滤是指入侵检测系统采用动态路由协议（BGP，OSPF等）和关键路由设备进行路由扩散，将需要过滤的IP地址扩散到路由设备中的路由表中，特点是响应时间快、容量大，但是只能动态地写入路由设备内存（RAM）中的路由表中，是易失的，同样粒度大。

连接欺骗指信息入侵检测系统在敏感连接传输过程中伪造连接结束信令（RST，FIN）发送给连接的源和目的地址，以中断该连接。特点是实时性强、粒度小（连接级），可以针对某一次敏感连接进行阻断。缺点是对分析系统工作状态依赖较强，需要向业务网上发送数据包，易受DoS攻击。

通过和连接级防火墙设备进行联动，可以针对连接五元组（传输协议类型、源地址、源端口、目的地址、目的端口）对数据流进行过滤。可以针对指定的任意五元以内的组合条件进行过滤，实时性强、粒度小。后来又加上了DNS劫持/污染，不过这个手动设置的机制已经不能算一个入侵检测系统的响应了。

日志记录

GFW有日志。这意味着什么？这就意味着当你芬兰国的时候，你的所作所为都记录在案。不光是你一个人，其他所有人都经常芬兰国。但据统计87.53%的人（361之316）都是无意之中芬兰国，从统计理论上看，记录在案的无效信息过多会造成信息难以利用。因此GFW后期一直在做“数据融合、聚类、分类的研究”，鸭子硬上弓，各种神经网络、概率模型、人工智能的论文整了一大堆，效果如何呢？

GFW的日志应该会记录这样一些事件信息：起始时间、结束时间、源地址、目标地址、目标端口、服务类型、敏感类型。[null]信息难以利用不等于不能利用，如果日志被翻出来了而且用户没有用代理，那么根据常识，从IP地址对应到人也只是时间问题。这就是说，GFW即使不能阻断，最差也是一个巨型监听设备。

规模估计

问题：　GFW的软硬件配置？

事实：　“虚拟计算环境实验床”是由国家计算机网络应急技术处理协调中心（CNCERT/CC）和哈尔滨工业大学（HIT）协作建设，以国家计算机网络应急技术处理协调中心遍布全国31个省份的网络基础设施及计算资源为基础，对分布自治资源进行集成和综合利用，构建起的一个开放、安全、动态、可控的大规模虚拟计算环境实验平台，研究并验证虚拟计算环境聚合与协同机理。2005年此平台配置如下：[05b]

CNCERT/CC	北京	曙光4000L	128节点	2*Xeon 2.4G	RAM2G
HIT	哈尔滨	曙光服务器	32节点	2*Xeon 2.4G	RAM2G
CNCERT/CC	上海	Beowulf集群	64节点	2*AMD Athlon 1.5G	RAM2G

结论：

GFW（北京）使用曙光4000L机群，操作系统Red Hat系列（从7.2[03a]到7.3[05a]到AS 4），周边软件见曙光4000L一般配置；GFW实验室（哈工大）使用曙光服务器[05b]，Red Hat系列；GFW（上海）使用Beowulf集群（攒的？）。

问题:

GFW与曙光是什么关系?

事实:

换句话说，是先有了用户的应用需求（蛋），才有了曙光4000L的研制（鸡）。这其实不难想像，一套价值几千万元的系统，如果纯是为了填补科学空白，将会延长产品市场化的时间。曙光4000L充分体现了中科院计算所在科研成果市场化方面的运作能力。……而曙光4000L这套系统就是针对国家信息化的实际应用而设计的。 曙光4000L的研制……曙光公司从事了工程任务和产品化工作，国防科技大学从事了机群数据库中间件的开发工作，哈尔滨工业大学开发了应用软件。 哈尔滨工业大学（威海）网络与信息安全技术研究中心日前成立，……方滨兴……揭牌。……曙光……向研究中心赠送了一套价值40万元的刀片服务器。

结论:

GFW是曙光4000L的主要需求来源、研究发起者、客户、股东、共同开发者。是不是应该打一点折? （曙光公司=中科院计算所）

问题:

GFW计算规模有多大?

事实:

2007年机群规模进一步扩大，北京增至360节点，上海增至128节点，哈尔滨增至64节点，共计552节点。机群间星型千兆互联。[null]计划节点数上千。[null] 曙光4000L……系统节点数为322节点，可扩展到640节点。根据功能的不同，曙光4000L可以分为服务节点、计算节点和数据库节点三类。每个计算节点2个2.4GHZ的Intel Xeon CPU，内存2GB。 2005年国家计算机网络与信息安全管理中心（北京）就已经建立了一套384*16节点的集群用于网络内容过滤（005工程）和短信过滤（016工程）。[来源不可靠] 64个节点、128个处理器（主频为2.8GHz）的曙光4000L……包括系统软件、管理软件、输入输出设备和存储设备，采购金额近千万。 才有了曙光4000L的研制……一套价值几千万元的系统。 国家信息安全重大项目“国家信息安全管理系统”（005工程）经费4.9亿。

猜测:

GFW（北京）拥有16套曙光4000L，每套384节点，其中24个服务和数据库节点，360个计算节点。每套价格约两千万到三千万，占005工程经费的主要部分。有3套（将）用于虚拟计算环境实验床，计千余节点。13套用于骨干网络过滤。总计6144节点，12288CPU，12288GB内存，峰值计算速度48万亿次（定义不明，GFW不做浮点运算，2003年top500排名榜首地球模拟器5120个CPU）。

问题:

GFW吞吐量有多大?

事实:

2GHz CPU的主机Linux操作系统下可达到600Kpps以上的捕包率。通过骨干网实验，配置16个数据流总线即可以线速处理八路OC48接口网络数据。[03b] 曙光4000L单结点的接入能力为每秒65万数据包，整个系统能够满足32Gbp的实时数据流的并发接入要求。

猜测:

512Gbps（北京）。

结论

噫吁嚱！危乎高哉！……

注释

null引用有特殊含义。

[*] 因为性能要求，负载平衡的完整算法必然很简单，不过我们一下子也没有猜出来。由于这个算法是易变的，即使猜出来公布在这里就立刻失效了，因此也没有在这个方向再费精力。

[**] 顺便指出论文中存在的一种硬伤。论文中反复把libpcap当反面教材作为性能低下的代表，称其是“传统TCP/IP栈之上的用户层函数库”“基于传统TCP/IP栈的libpcap”。可是人家libpcap从2001年1月的0.6版本就开始用2.2以上版本内核提供的packet(7) socket，这个跟TCP/IP一点关系都没有。怪罪的对象错了，要怪的是packet(7)而不是libpcap。后来2004年PF_RING出来，设计很相似，libpcap用上一样nb。不过这个时候GFW也已经研发完了。

[***] 如果将其视为bug而不是feature的话，漏洞实在太多，打一两个补丁不解决问题，非重做不可。另外IP碎片和TCP流重组没有做特别研究，即使有漏洞实用性也不会很高。

[03a] 杨武, 方滨兴, 云晓春, 张宏莉. 基于骨干网的并行集群入侵检测系统. *哈尔滨工业大学学报*. 2003-5-15.

[03b] 陈训逊, 方滨兴, 李蕾. 高速网络环境下入侵检测系统结构研究. *计算机研究与发展*. 2003-7-15.

[05a] 张兆心, 方滨兴, 胡铭曾. 支持IDS的高速网络信息获取体系结构. *北京邮电大学学报*. 2005-2-25.