Virus Detection System

—— 网络病毒监控系统的架构体系与研究方法

肖新光 1 吴冰 2 云晓春 3 邱永良 4

（1、4：中国安天实验室 哈尔滨 150006；2、3：哈尔滨工业大学 哈尔滨 150006）

摘要

通过对传统的 IDS 结构进行分析，研究者认为传统的 IDS 的体系结构不适合对高速大流量下的种类繁多的病毒进行检测。为解决这个问题，研究者以准确而高速的检测各种病毒在网络中传输、扫描、攻击行为并兼有部分未知病毒检测能力为目标。提出了基于旁路监听技术，采用归一化处理思想，面向算法效率的检测系统体系结构——Virus Detection System（VDS)；，实现了适用于骨干网络带宽条件，采用简单协议分流，并发大特征集高速匹配处理和并发协议定位解析的检测体制。论文通过引入 AVML、DEDL 等新的技术描述体制论述了 VDS 使用的数据处理方式，并介绍了深度处理和未知发现的一般性方法。

关键词：计算机网络，安全，病毒，归一化方法，Virus Detection System

简介

根据 Antiy Cert 的统计，2004 年全球病毒产生速度加快，其中与网络相关的蠕虫、后门、木马数量均增量增长，全年共产生新病毒 20047 种，比 03 年增长 41%，超过了 1986-1996 的十年总和，从而使全球病毒总数达到 7 万 3 千种。

由于网络相关的病毒对网络基础设施、信息系统环境压力和安全影响日趋加大，理论和工程界都在不断的提出和修正有关的方法，Virus Detection System（以下简称 VDS）就是本文作者所提出的一套体系化方法和对应架构。

IDS 向 VDS 的体制变化原因

1、IDS 体制是 VDS 体制的基础

VDS 与 IDS（此处指 NIDS）系统设计的基本思路相同，都是基于若干检测规则，实现不影响网络效率的检测与数据处理。因此 VDS 采用了 NIDS 的基本体制，选择了旁路监听方式。即采用将网卡设置为混杂模式接入共享器（hub）端口或者接入交换机（switch）镜像端口的方式实现网络数据的实时旁路捕获，并对所获得的网络数据进行进一步的检测处理。

2、传统 IDS 体系架构

传统的网络 IDS 主要采用基于精细的协议解析分流网络数据后，通过若干个小特征库的匹配的体系架构。图 1 为传统的网络 IDS 模型。捕获层首先捕获所有的网络通信内容，并将收到的网络数据进行协议解析，解析出协议头部的各个域，然后据此进行数据分流，将数据送入检测引擎进行对应协议的规则子集检测。

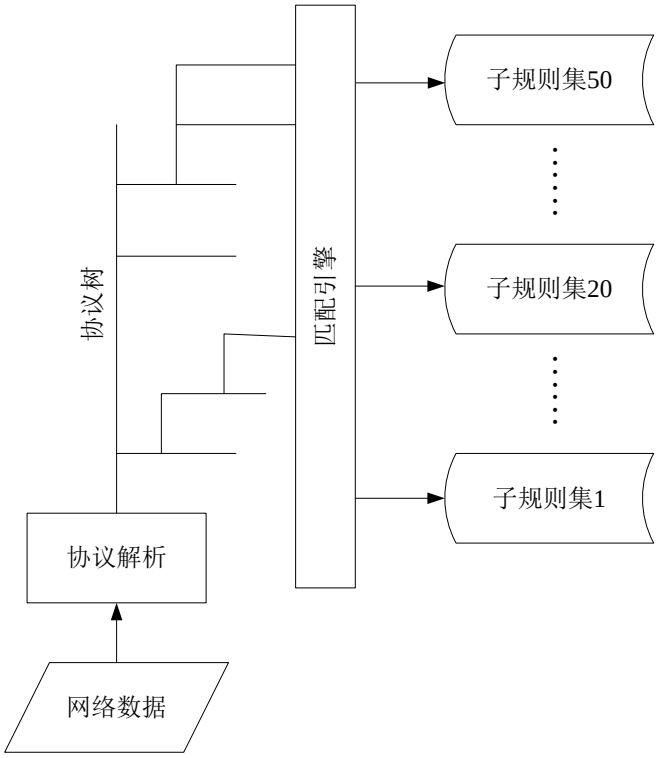
这里我们用归一化软件结构方法来解释传统 IDS 模型。

定义一：归一化方法，在面对大规模复杂事件处理的情况下，采取的对需要处理的事件

做归类处理,形成一个或一组的相应的处理模块和对应的可扩展数据结构与数据集合的规划方法。

传统网络 IDS 的归一化方法是通过精细协议解析后，进行小规则集匹配。由于绝大多数网络攻击行为与具体网络协议和服务存在依存关系,因此对脱离了网络协议的特征描述可以视为无效规则，所以精细协议解析有效的将待检测对象分流，避免无效规则检测。

这种方法有效缩小了规则集的规模，提高了检测效率。在 IDS 系统发展的初始阶段，影响系统效率的瓶颈问题一直是匹配速度。由于 IDS 检测算法的时间长度与纪录条数呈线性关系，因此获取最短检测时间的有效方式是最大程度的减少特征纪录的条数。通过精细协议解析实现小规模的特征库匹配,提高了检测速度。因此当前多数网络 IDS 都是采用与 snort 相似的架构：将数千条规则分配到几十个小型的按照协议归一化相关的特征库中。所以 IDS 的检测速度和检测粒度取决于三个主要因素：协议解析的深度、特征匹配的速度、特征库的质量。其中检测的速度与协议解析的深度呈反比，而与特征匹配的速度成正比。



图一：传统 IDS 体制

3、网络病毒全面检测目标与传统的 IDS 体制的冲突

全面病毒检测应该包含如下的对象检测：

行为	说明
蠕虫主动传播	包括邮件蠕虫、口令猜测蠕虫等的主动发送自身的行为
病毒体传输	攻击者主动传递病毒的行为和染毒文件被下载等行为
病毒扫描	病毒所发出的扫描包和溢出包
病毒攻击	病毒感染节点发动的各种 DoS、DDoS 等攻击
病毒在线升级	具有 online update 能力的蠕虫的在线升级行为。
病毒握手连接	后门的握手过程

其他病毒行为	如连接 IRC，接受控制指令等

表一：期望检测的病毒行为

为了更好的表示病毒检测的方法，我们引入 AVML 概念。

定义二：AVML ，AVML 是 AntiVirus Markup Language（病毒检测规则描述语言 Virus detection rule scription language）的缩写，是一种文本化的描述病毒检测与处理规则的方法。

AVML 由 Antiy Labs 在与哈尔滨工业大学的公益性研究计划 OBAV（Open Base Antivirus）计划中提出，在此前有关论文中也将 AVML 称为 VCC(Virus Characteristic Characterization)。

下面示例为通过 AVML 表示的 Backdoor.Win32.bo.a 后门文件的检测方式。

Echo

irus(id=”B00801”;type=”Backdoor”;os=”Win32”;format=”pe”;name=”bo”;version=”a”;size=”124928”;Port_listen=on[31337];content=|81EC0805000083BC240C050000005356 57557D148B8424240500008BAC242005000050E9950500000F85800500008B|;delmark=1)

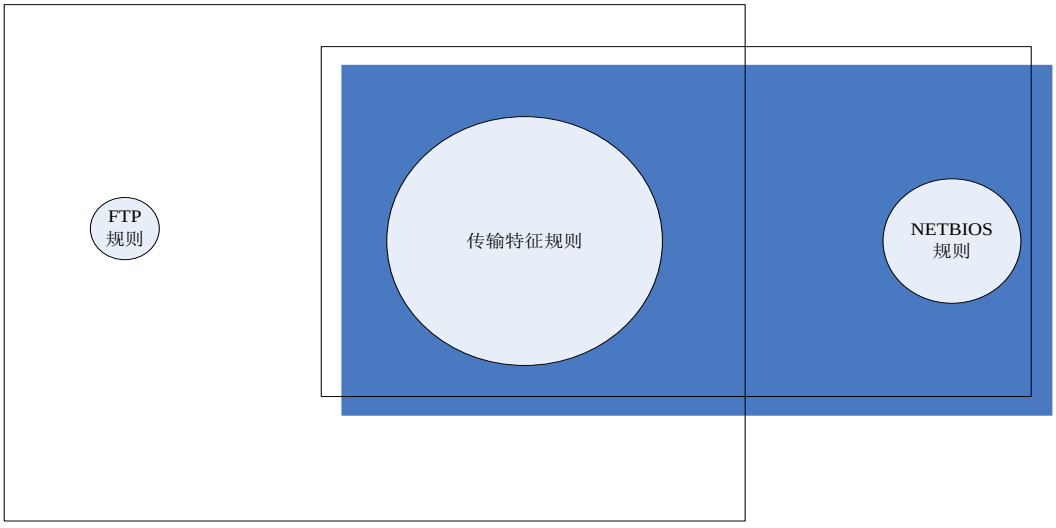
假定需要检测 Backdoor.bo 通过 FTP 服务上载，可以把 AVML 描述转化为类似 Snort 风格的 IDS 规则。

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Backdoor.bo.a Upload"; content: |81EC0805000083BC240C05000000535657557D148B8424240500008BAC242005000050E9950 500000F85800500008B |;)

如果需要检测 Backdoor.bo 通过 NETBIOS 服务进行自身复制，则对应的 Snort 风格规则如下，

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"Backdoor.bo.a Copy"; content: |81EC0805000083BC240C05000000535657557D148B8424240500008BAC242005000050E9950 500000F85800500008B |;)。

可以发现，除了系统前端的分流机制的变化，两条规则的检测方式是一致的，如果采用 snort 的分流体制，并期望与原有规则的整合，则规则集将出现冗余（如图二），由于病毒规则集的庞大，冗余部分远大于原有 IDS 规则规模。如果保留原有的 snort 体制，把病毒规则检测作为一个公共前置或者后置部分，则检测过程出现冗余。同时网络协议中存在多种文件传输方式，而一些传输方式使用的端口则是随机的，这也给传统分流机制带来问题。



图二：大量公用的传输特征规则造成传统体制的冗余

体制冗余只是一种表观问题，如果引入传输特征测试，网络病毒监测面临更大的问题是规则的规模压力。根据 ASTS#6 基础平台的统计，截止到 2005 年 7 月 1 日，全球各种网络蠕虫累计数量已经达到 5376 种（如表二），与网络相关的木马、后门、黑客工具、间谍软件的总数累计也已经超过 3 万种。

分类	数量
邮件蠕虫	2807
即时消息蠕虫	172
P2P 环境蠕虫	1007
IRC 蠕虫	715
其他蠕虫	675
总计	5376

表二：截止到 2005 年 7 月 1 日的蠕虫分类数量表

由于病毒名称与病毒检测规则可能是一对多映射，因此需要的检测规则数量比病毒种类数更大，需要数万条条类似的规则，才能够全面检测目前所有具有主流操作系统活性的病毒。

为此早期 IDS 所采用的记录条数与时间呈现线形关系的算法，将因性能严重衰减而不可用。如表三所表示：

算法对比				
特征串的个数	100	200	500	1000
一般字符串查找的匹配时间	0. 30	0. 6	1.3	2.5
有限自动机的匹配时间(s)	0.01	0.03	0.07	0.18
AVL 引擎匹配的时间(s)	0.01	0.01	0.02	0.03

表三：实验数据对比

引自：哈尔滨工业大学国家内容安全重点实验室 2002 年 9 月测试数据

该表显示了一般性字符串查找方法、有限自动机算法(snort 采用的算法)、研究者目前所采用的 AVL 引擎（V1.0）之间在特征串增长上呈现的关系。从以上的结果可以看出，传统字符串查找算法效率很低，在规则较多的条件下失去了使用价值。对于相同的文本长度，当模式串的数量小于 100 时，AVL 引擎的匹配时间与自动机算法的效率差不多。而当模式串的数量越多时，后者的匹配速度比前者明显加快。

VDS 体制

1、核心架构

鉴于网络病毒检测系统可用性的关键在于解决超大规模规则集检测问题，因此研究者选择探索一个异于传统 NIDS 的新体制。这个新体制的归一化模型以匹配速度和匹配粒度为核心，其系统结构是面向匹配算法的，而不是面向协议分析的。我们把这套体制称为 Virus Dection System,即 VDS。

基于算法设计检测模型，我们将需要进行内容匹配网络数据划分为以下三种类型：一是可以直接进行 2 进制级别匹配的数据，二是需要深度内容预处理的数据，三是需要特定算法

的数据。

需要深度内容预处理的数据主要针对两类数据：一是对于 WEB 页面，为了能够更细粒度的检测脚本病毒，需要实现一个面向连接的脚本压缩器，以便能在网络级别对网络脚本病毒进行实时监测；二是需要针对经过编码的邮件进行动态预处理，作为实现邮件病毒的识别的前导模块。传统 IDS 很难实现上述两种情况的检测。需要特定算法的数据是指针对如 URL 攻击等大小写不敏感的数据，实现一个与大小写不敏感的算法。

所以我们提出新的归一化思想是除上述三种外的所有网络数据都视为可以直接进行二进制级别匹配的数据。新的归一化模型中省略了精细的协议树解析，只保留了一个最基本的数据分流体制。同时考虑到一些网络行为并不依赖于其特征，而只对协议有依存关系。同时考虑到降低误报的可能，一些特征需要相关的网络信息辅助验证。因此 VDS 最终形成了如图所示的简易网络分流-大规则集检测—辅助网络验证的体制，并形成了了如下的规则类型。

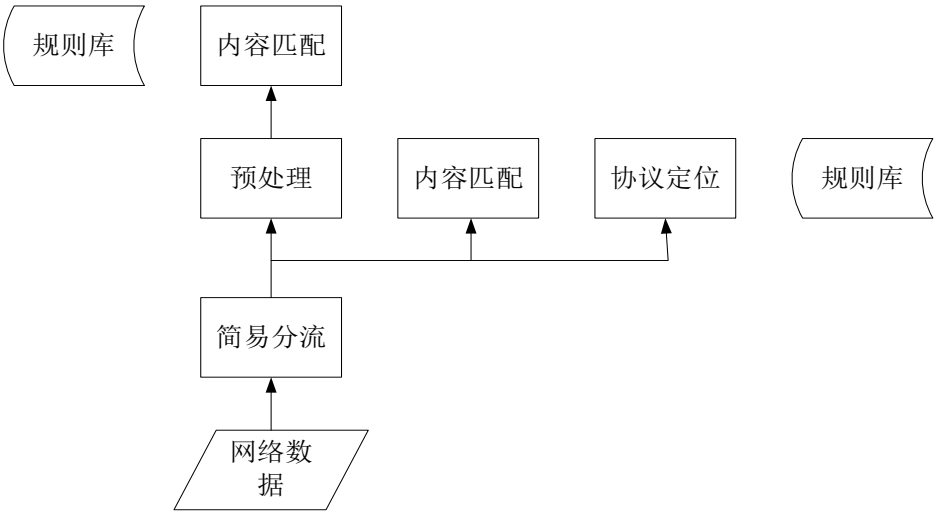
（1）基于深度内容预处理的匹配规则

（2）基于特定算法的匹配规则

（3）进制级别匹配的规则

（4）进制级别匹配需要网络信息校验的规则

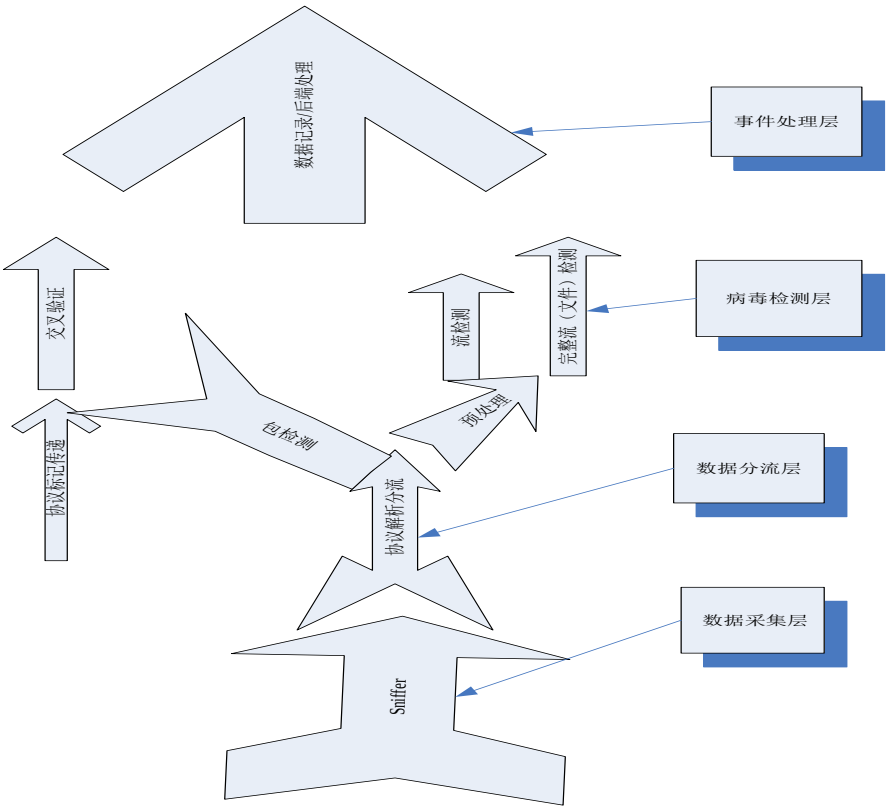
（5）纯网络信息规则



图三：VDS 核心思想的变化（对照图一）

2、病毒检测的实现层次

考虑到兼顾病毒检测效率和粒度问题，研究者在 VDS 体制中尝试提供包、非完整流、和完整流（文件）三层次的病毒检测（如图 4），其中包级和流级的病毒检测为高速检测，完整流级别的检测是采用传统文件级引擎的慢速检测，相关细粒度可嵌入反病毒引擎的思想，研究者在 xfocus2004 中已经介绍过，此处不给与更多说明。

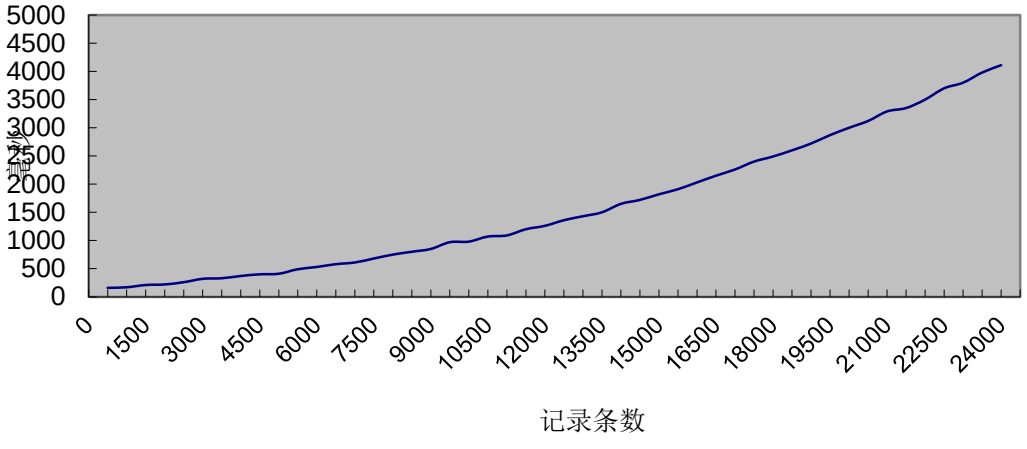


图四：VDS 数据流传过程和检测层次

在已知病毒检测的同时，研究者通过 VDS 架构尝试实现了 3 种方式的未知病毒检测，即基于神经网络的可疑脚本发现、基于短广谱特征的加权发现和行为归纳发现三种。

3、引擎算法优化

随着特征的增多，建立在一般记录与匹配时间非线性算法基础上的高速引擎也会呈现出严重的问题，如图五：

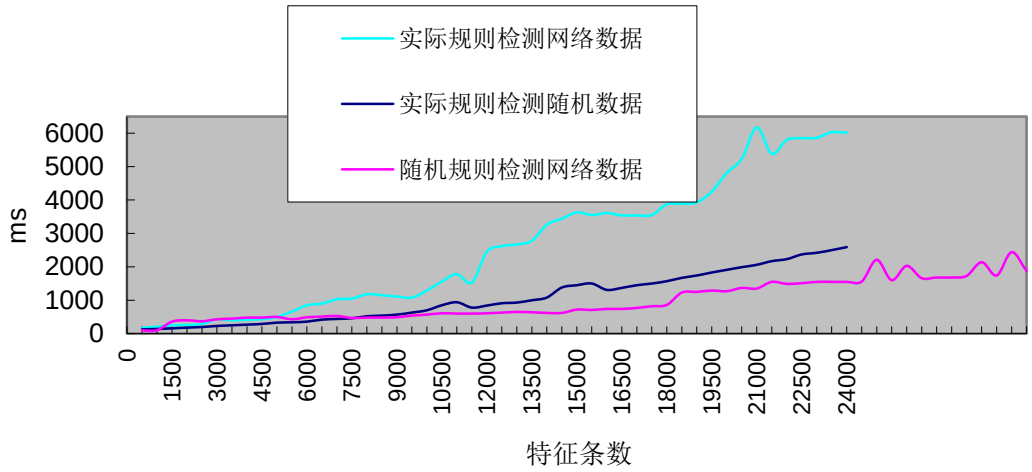


图五：数据匹配时间随记录条数的变化（匹配对象为随机生成）

该测试反映了检测引擎对于同一检测对象数据，在每增加 500 条特征情况下测试一次的时间表现，呈现出高速匹配算法缺乏特定优化条件下，所呈现的与记录条数与匹配时间对照关系。在特征规则小于 6000 条的情况下， 时间与记录条数的线性增长关系并不明显。但在

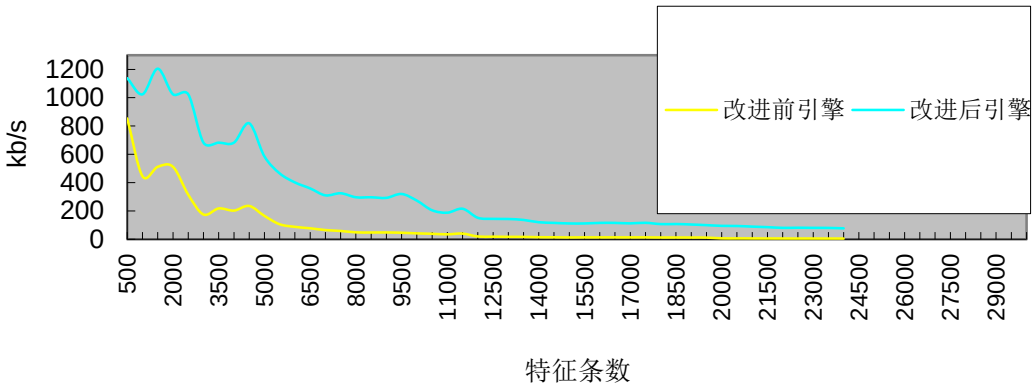
10000 条记录左右，开始出现反向的越升，导致性能的急剧下降，直至不可用。

同时，检测速度也与匹配对象和模式串质量存在密切关系，由于病毒特征之间存在着一定的近似性，并不能呈现随机分布的特点。同时网络数据也呈现出一定的分布特性，也并不呈现随机分布的特点。因此都对匹配情况构成影响，图六表示了这种影响。可见在检测特征较多的情况下，采用实际规则检测网络数据的实际引擎工作环境（即 VDS 的实际工作条件），具有最差的性能。



图六：针对不同规则和不同检测数据检测时间与规则条数的关系图表

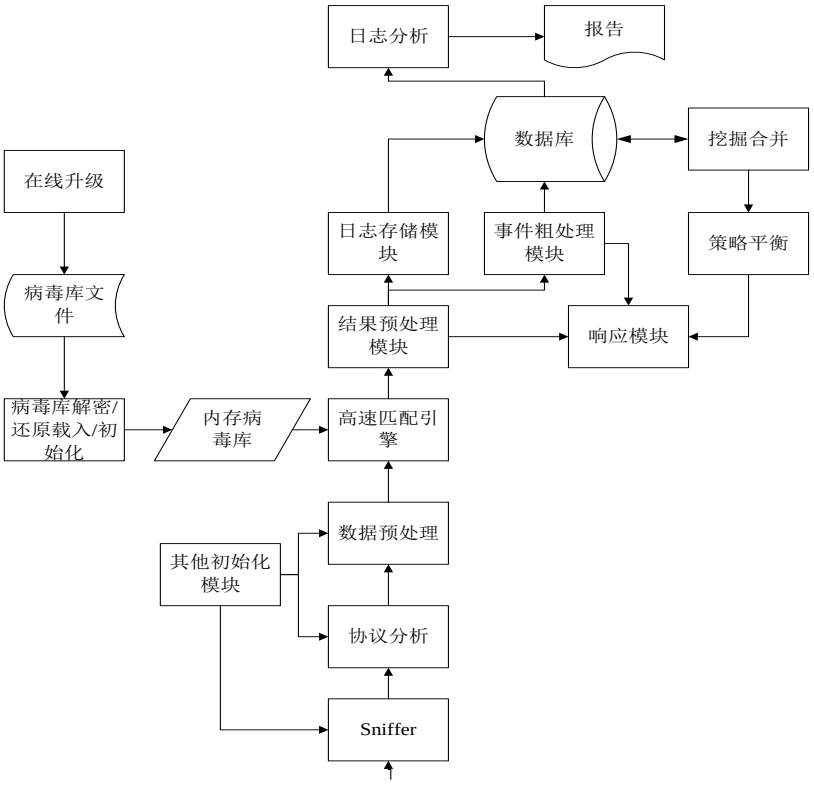
这种基本的算法能力对 VDS 能力的制约是明显的，因此对这种算法实施改进和优化。我们的基本思路是遏制病毒特征码近似性带来的效率影响。图 7 演示了通过近似性特征优化的引擎与优化前的效率对比。



图七：算法优化对匹配速度的影响

4、实现架构尝试

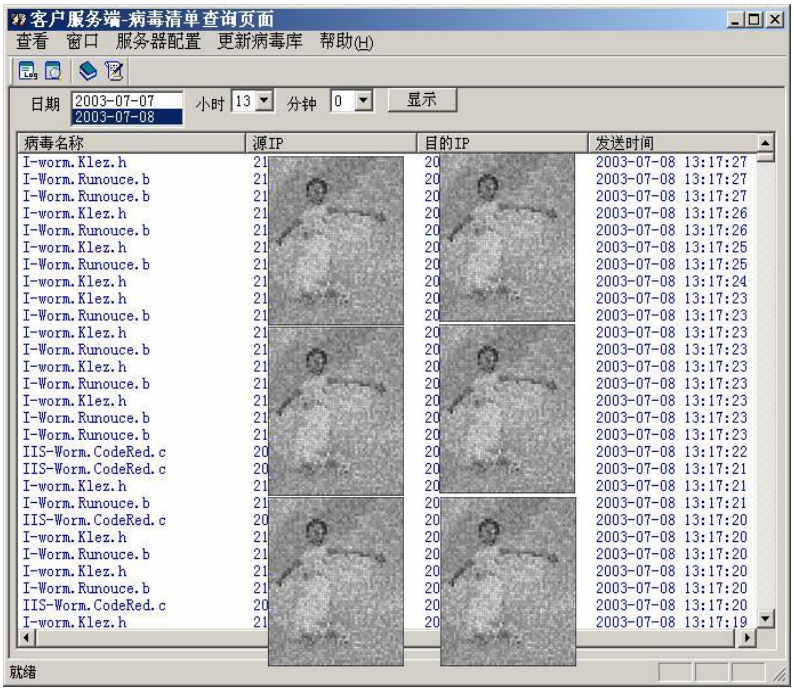
VDS 系统结构图见图八，系统中 Sniffer 为网络数据捕获部分，采用 Zero Copy 技术。



图八：VDS 系统结构设计图

5、测试与实验

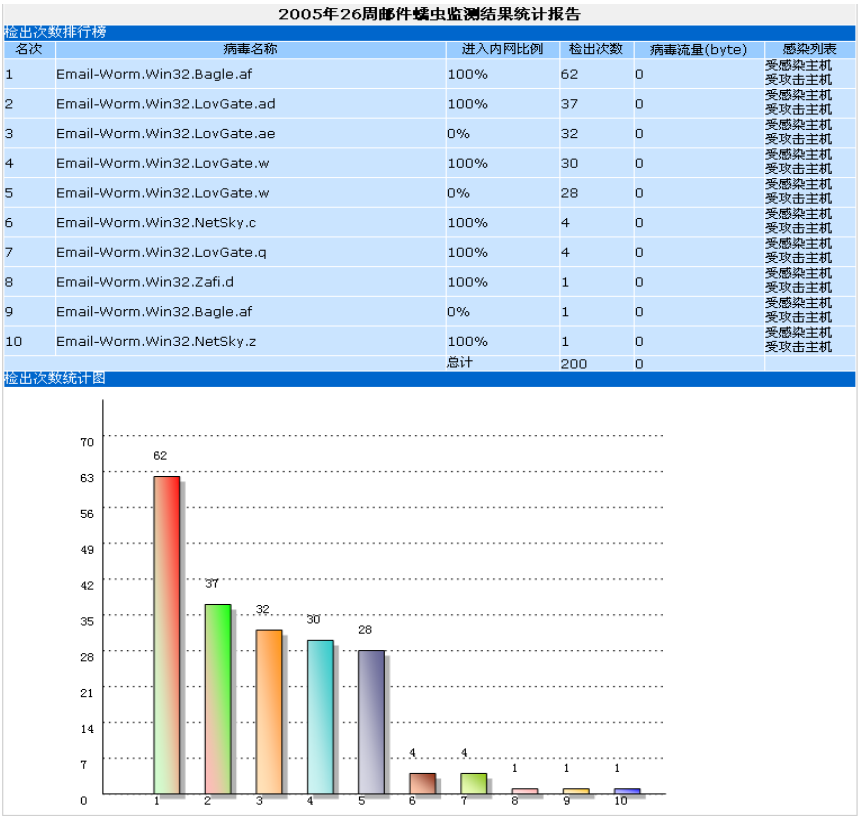
1) 实验系统于 2002 年 12 月开始在哈尔滨工业大学出口上投入试运行，出口带宽为 1000M。



图九：VDS 客户端 实时病毒事件

2003 年 7 月 8 日哈尔滨工业大学出口实时病毒数据。

2) 实验系统从 2003 年 3 月起开始向国家有关部门上报有关网络病毒的每日状况，日常报表类似下图。



2005年26周邮件蠕虫监测结果统计报告

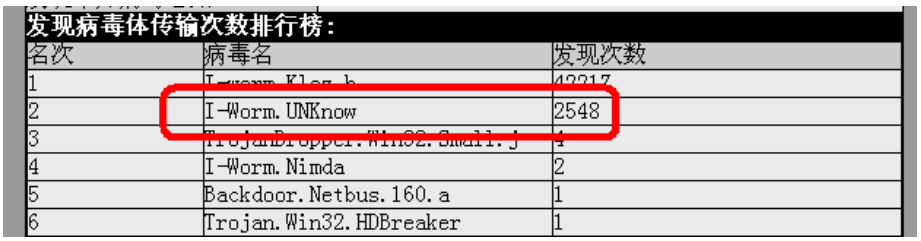
检出次数排行榜

名次	病毒名称	进入内网比例	检出次数	病毒流量(byte)	感染列表
1	Email-Worm.Win32.Bagle.af	100%	62	0	受感染主机 受攻击主机
2	Email-Worm.Win32.LovGate.ad	100%	37	0	受感染主机 受攻击主机
3	Email-Worm.Win32.LovGate.ae	0%	32	0	受感染主机 受攻击主机
4	Email-Worm.Win32.LovGate.w	100%	30	0	受感染主机 受攻击主机
5	Email-Worm.Win32.LovGate.w	0%	28	0	受感染主机 受攻击主机
6	Email-Worm.Win32.NetSky.c	100%	4	0	受感染主机 受攻击主机
7	Email-Worm.Win32.LovGate.q	100%	4	0	受感染主机 受攻击主机
8	Email-Worm.Win32.Zafi.d	100%	1	0	受感染主机 受攻击主机
9	Email-Worm.Win32.Bagle.af	0%	1	0	受感染主机 受攻击主机
10	Email-Worm.Win32.NetSky.z	100%	1	0	受感染主机 受攻击主机
		总计	200	0	

检出次数统计图

图十：2005 年第 26 周邮件蠕虫检测结果统计报告（哈尔滨工业大学出口数据）

3) 病毒预警：VDS 预警了多个蠕虫。下图是 VDS 预警 sobig.f 蠕虫的截图。



发现病毒体传输次数排行榜：

名次	病毒名	发现次数
1	I-Worm.Klez.h	42217
2	I-Worm.UNKnow	2548
3	TrojanDropper.Win32.Small.j	4
4	I-Worm.Nimda	2
5	Backdoor.Netbus.160.a	1
6	Trojan.Win32.HDBreaker	1

图十一： 2003 年 6 月 5 日，黑龙江教育网主节点统计数据。

6 月 5 日发现 I-Worm.Unknow（未知邮件蠕虫）数量显著增长，6 月 6 日证明，该 I-WORM.Unknow 为 I-worm.sobig.f。

数据处理方法

由于蠕虫事件已经占据网络安全事件主流，因此 VDS 面临比 IDS 更为严重的记录淹没问题。因此 VDS 采用了一系列的数据处理方法。为了良好的描述数据处理，我们引入 DEDL 的概念：

定义三：DEDL，DEDL（Detection Event Description Language），即检测事件描述语言。该方法是采用描述符的方式，将网络检测事件制定为一种规范的格式，并支持一般的条件推导的一种安全事件规范化描述方法。

DEDL 定义了事件类型（type）、事件 ID、源 IP（Source_IP）、目标 IP(Target IP)、事件时间等 20 多个事件要素。

VDS 所应用的基础数据处理方式

为了避免记录淹没效应，同时表示检测记录的关联关系，VDS 内置了六种事件处理方式。

a. 内部技术合并：一般是指系统的内部处理，对用户透明，主要是由于传统反病毒引擎是采用双特征码确认的机制检测，为了保证不误报为两条记录，需要内部合并。

b. 平行式合并：平行式合并是指在把多条 Source_IP、 Target IP 相同、Type 和 ID 都相同的情况下的记录合并为一个事件。比较典型的是感染邮件蠕虫计算机的邮件蠕虫发送事件，和邮件服务器之间的蠕虫发送事件；

c. 分析式平行合并：分析式平行合并是指对 Source_IP、 Target_IP 相同但是 Type 和 ID 不同的事件进行合并处理。比较典型的是点对点扫描事件；

d. 辐射式合并：辐射式合并是指把多条 Source_IP 相同 Target IP 不同的记录合并，一般是指 Target IP 能形成连续的 IP 范围，而 Target Port 相同，Type 和 ID 都相同的情况的纪录。比较典型的是蠕虫和黑客的区段扫描行为；

e. 聚合式合并：聚合是合并是指把多条 Source IP 不同，但 Target IP(包括 Target Port）相同的记录合并。比较典型的是 DDoS 攻击行为、蠕虫的 Online Update 行为和连接 IRC 的行为。聚合式合并是 VDS 独有的处理方式。

f. 传导链式合并：传导链式是指按照感染关系形成 IP 链条，我们对于 IDS 的传导链方法做了一定修正。

目前 IDS 系统的传导链式合并基本是以事件名称一致的，较早事件的 Target IP 和较晚事件的 Source_IP 一致为准：

```
If existNet_Action(RPC_Exploit)[IP(1)->IP(2);time(1)]
Net_Action(RPC_Exploit) [IP(2)->IP(3) ;time(2)]
and
time(2)>time(1)

than
Net_Action(RPC_Exploit) [IP(1)-> IP(2) -> IP(3)]
```

例 1：传统 IDS 传导链的 DEDL 描述（简化表示）

这个传导链需要假定探头具有的充分的覆盖能力，而没有考虑到先期感染节点再次感染内部节点的过程没有被监控到的可能。

而 VDS 系统的传导链，可以支持内网感染的猜测。

```
If exist
Net_Action(Trans,Worm.Win32.Dvldr)[IP(1)->IP(2);time(1)]
Net_Action(Trans,Worm.Win32.Dvldr)[IP(3)->IP(4);time(2)]
```

and
NET(a) ∈{IP(2), IP(3)}/IP2,IP 属于内网 NET(a)
and
time(2)>time(1)
than
Net_Action(Trans,Worm.Win32.Dvldr) [IP(1)-> IP(2) -> IP(3) ->IP(4)]

例 2：VDS 传导链修正方法的 DEDL 描述（简化表示）

深度数据处理与未知发现

1、行为归并

VDS 基于大特征集通过检测蠕虫的传输特征实现了精确的病毒检测，同时也检测蠕虫相关的网络行为，但难点在于将蠕虫传输事件与网络行为建立关联，否则系统仍无法解释网络行为是否因病毒感染所导致。从技术手段上来讲，如果没有主机环节配合并无技术上直接推理的方法。因此研究者通过在病毒库中放置更多的标志的方法来实现。

这里通过 DEDL 与 AVML 的行为描述体现特征检测的冲销关系。

DEDL 记录	AVML 行为特性规则
Net_Action(act)[IP(1),IP(2):445; ;time(1)] *Net_Action(act)[IP(1),IP(3):445; ;time(1)]* *….* *Net_Action(act)[IP(1),IP(12):445; ;time(1)]* *Net_Action(Trans,Worm.Win32.Dvldr)* *[IP(1)->IP(12);time(1)]*	*Virus_act_lib* *Virus seek(id=”W02872”;dport=139,445* *;trans=netbios)*

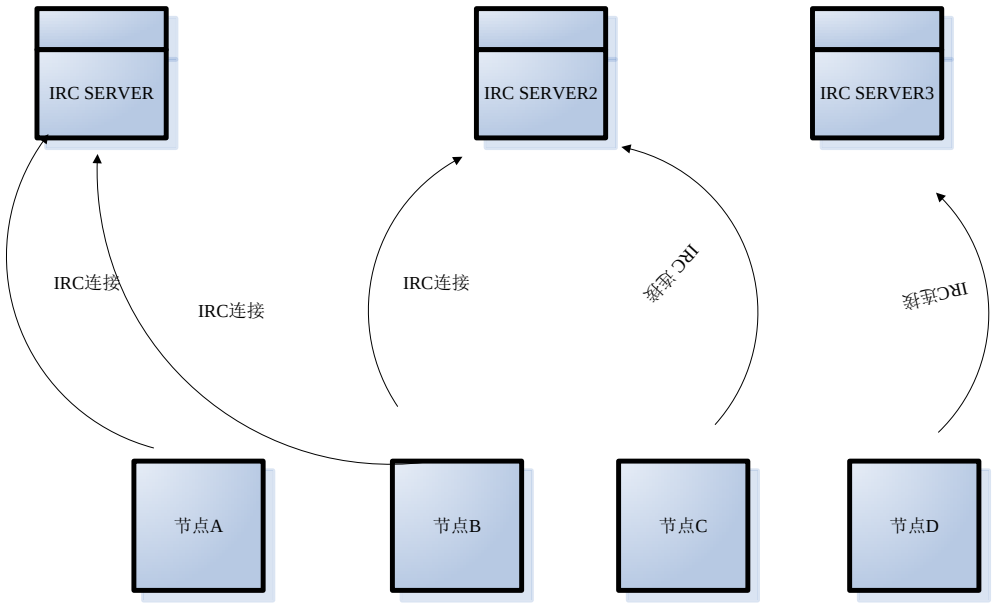
例 3 Worm.Win32.Dvld 的 DEDL 事件与 AVML 针对 Worm.Win32.Dvld 和 445 端口访问进行的冲销行为

就此可以将 IP（1）的 445 扫描事件与 Worm.Win32.Dvldr 的传输事件合并。

2、数据处理式发现

VDS 的平行、辐射和聚合处理即是避免形成记录淹没的方法，也是实现未知检测的方法。这里以辐射式合并为例。

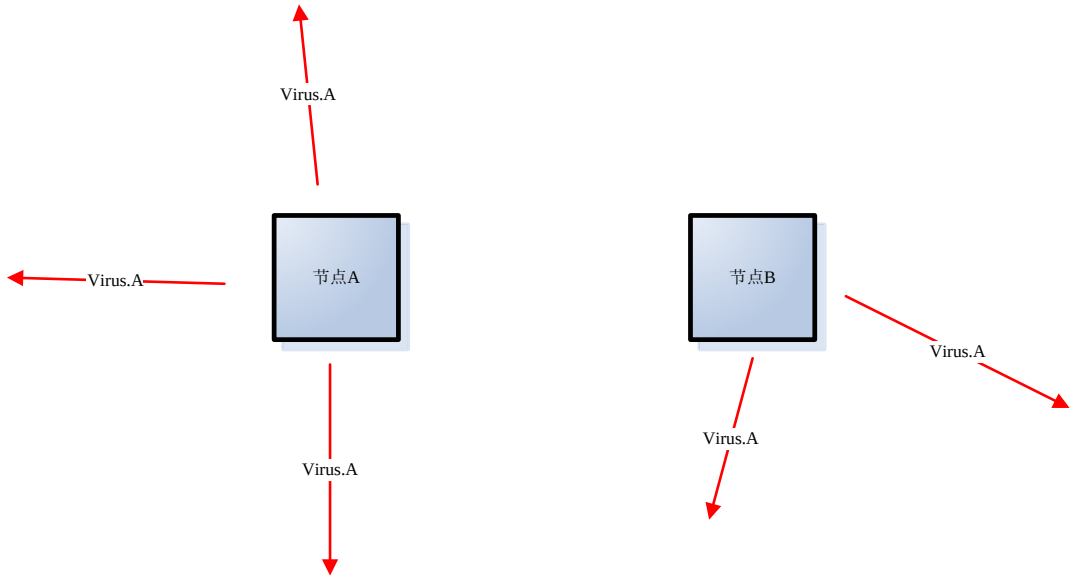
假定依据事件形成的应用拓扑图表如图十三，假定 IRC 为敏感行为，且形成非正常聚合态。则可认定节点 A、B、C 为可疑节点。



图十二：一个组网络行为示意图

3、基于行为归并和数据处理的发现

由于网络上的实际情况是有漏洞的系统会感染多种病毒，所以简单的行为归并，可能会使其他一些病毒的事件被隐藏，因此我们将行为归并和数据处理进行结合，假定如图十四的行为检测拓扑。那么，节点 B 只冲消掉了通往 IRC_SERVER 的行为，而保留了通往 IRC_SERVER2 的行为。从而认为向 IRC SERVER2 的聚合是可疑行为。



图十三：病毒传输事件图 与图十二进行联合分析的一个特征检测视图

结束语

VDS 的体制研究是网络病毒监控多年的学术化探索和工程化尝试的一个聚合。

VDS 与一般性的网络蠕虫学术研究方法的进步在于，传统的学术方法立足于基于网络

行为和流量变异的未知发现。但未知本身就是相对已知而言，没有精确的已知检测能力为基础，也就失去了未知检测的意义。

由于VDS其引入的海量的病毒精确规则检测,因此将基于宽带甚至骨干的准确到病毒名称病毒检测成为可能。并以此为基础形成一套研究方法。

仅以此文献给 4 年来，为骨干网病毒监控课题此做出努力的伙伴们和给与我们支持和关心的朋友们。

参考文献：

[1] Wu Bing,Yun, Xiaochun, Xiao Xinguang. “Backbone Network Worm Pressure Measurement System Based on Bypass Monitor,” *AVER,2004*

[2] Wu Bing,Yun, Xiaochun, Xiao Xinguang., “VDS: Malcode Detection System,” *ACM,WORM 2005.*

[3] Wang Bailing, Fang Binxing, Yun Xiaochun, "The Study and Implementation of Zero-Copy Packet Capture Platform," *Chinese Jouranl of Computers, Vol.28, No.1, Jan., 2005*.

[4] Xiao Xinguang(seak), “细粒度可嵌入的反病毒引擎” *Xcon，2004*

[5] Xiao Xinguang(seak), "基于网络流和包的病毒检测.” *Xcon，2002*