Virus Detection System – the Architecture of and Approach to A Network Virus Detection System

Xinguang Xiao[1] Bing Wu[2] Xiaochun Yun[3] Yongliang Qiu[4]

1,4 Antiy Labs   2,3 Harbin Institute of Technology

Abstract:

After detailed analysis of the structure of traditional IDS, we believe that it doesn’t work well on a high speed network with various viruses. What we need is accurate and high-speed detection of virus transmission and of the attacks of known viruses as well as as-yet-unkown viruses. Based on port mirroring and the normalization theory, we developed the algorithm efficiency oriented Virus Detection System (VDS). We implemented a mechanism which adapts to the bandwidth by adopting simple divergence, a parallel large sign set and high speed matching, and parallel protocol resolution. This paper presents the data processing method used by VDS by introducing the AVML and DEDL. Deep processing and unknown virus discovery are also introduced.

Keywords

computer network, security, virus, normalization, Virus Detection System.

1. Introduction

According to statistics by Antiy CERT (2004), the number of viruses all over the globe is increasing faster and faster. Among them, worms, backdoors, Trojans, and other network-related malware, all spread more and more rapidly. In the whole year, about 20047 new viruses came out, which represents a 41 percent increase over 2003 and is greater than the sum of all viruses released from 1986 to 1996, bringing the total number of viruses to 73,000.

Because network related viruses have more and more influence on the network infrastructure and information system, both researching and engineering fields are proposing and modifying related methods. The Virus Detection System (VDS) is proposed by us as a system of methods and a corresponding architecture.

2. Why move from IDS to VDS

2.1 IDS is the basis of VDS

VDS and IDS (specifically NIDS) have the same design patterns: based on some detection rules, implement detection and data processing without affecting the networking efficiency. VDS adopts the mechanism of NIDS, a network tap. That is to connect to a hub via a promiscuous network card or connecting to the mirror port of switch, and then conduct further detection routines on the data.

2.2 Traditional IDS architecture

The traditional NIDS basically uses deep protocol resolution to analyze the network flow by matching against several small characteristics sets. Figure 1 shows the traditional NIDS model. At first, the packet capturing level obtains all the network traffic, and then resolves the data according to protocol. After obtaining the fields in the protocol, redirect the traffic to the corresponding rule set.

Here, we interpret the traditional IDS model using software normalization methods.

Definition 1: Normalization methods, faced by large scale and complicated event handling, are necessary to classify events. This classification can then be used to form several corresponding process modules and an extendable data structure and data set classifying methods.

The normalization method of traditional NIDS is to match after accurate protocol resolution. However, since most network attack behavior corresponds to a specific protocol, those rules which have nothing to do with this protocol description can be regarded as ineffective rules. Accurate protocol resolution can help avoid ineffective rules.

This method reduces the scale of the rule set, and enhances the detection efficiency. In the early phase of IDS， the bottleneck which affected the system efficiency was the matching speed. Because the matching speed of IDS has a linear relation to the number of records, to reduce the number of records is to reduce the time consumed. Through accurate protocol resolution and small rules set matching detection speed can be increased. So far, most of the NIDS have adopted a similar architecture to Snort: adopt thousands of rules in some small rule sets.

So, the speed of IDS and the granularity of IDS are determined by three factors: the depth of protocol resolution, the speed of matching, and the quality of the rule sets. Among them, the detection speed is inversely proportional to the depth of protocol resolution, while directly proportional to the matching speed.

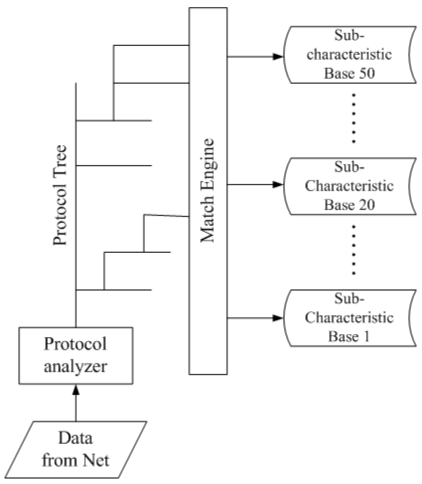

Sub-characteristic Base 50
Sub-Characteristic Base 20
Sub-Characteristic Base 1
Protocol Tree
Match Engine
Protocol analyzer
Data from Net

Figure 1：Traditional IDS mechanism

3.1 Conflicts between the virus detection goal and IDS mechanisms

Virus behavior detection includes following items：

Behavior	Description
Active spreading of worm	Including mail worms, password guessing worms sending themselves
Transportation of the virus body	Attacker actively spreading the virus, virus file is downloaded
Virus scanning	Scanning packets and exploit codes.
Virus attacking	DoS, DDoS attacking
Virus updating	Online update behavior of some worms
Backdoor opening	Opening backdoors for other viruses
Other virus behavior	Connecting to IRC, getting control commands

Table 1：Virus behaviors that should be detected

In order to better describe the virus detection methods, we bring in AVML

Definition 2: AVML, AntiVirus Markup Language, is a kind of method to describe virus detecting and processing.

AVML is proposed in the OBAV (Open Base Antivirus) plan. Some papers named it VCC (Virus Characteristic Characterization)

The following example shows how Backdoor.Win32.bo.a is detected via AVML:*Echo virus(id=”B00801”;type=”Backdoor”;os=”Win32”;format=”pe”;name=”bo”;version=”a”;size=”124928”;Port_listen=on[31337];content=|81EC0805000083BC240C050000005356557D148B8424240500008BAC242005000050E9950500000F85800500008B|;delmark=1)*

If it is necessary to detect the FTP uploading of Backdoor.bo, AVML can be converted to a Snort-like IDS rule.

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"Backdoor.bo.a Upload"; content: |81EC0805000083BC240C050000005356557D148B8424240500008BAC242005000050E9950500000F85800500008B |;)

If it is necessary to detect the spreading of Backdoor.bo via the NETBIOS service, its corresponding Snort rule is:
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"Backdoor.bo.a Copy"; content: |81EC0805000083BC240C05000000535657557D148B8424240500008BAC242005000050E9950 500000F85800500008B |;)

We can see that, except for the different methods used in front end, the two detection rules are the same. If AVML is integrated with the Snort architecture，the rule sets will have redundancy (figure 2）. For the large set of virus rules, the redundant part is much larger than all the entire former IDS rule sets. If the Snort mechanism is kept in place, with the virus detection rules added, there would be large redundancies in detection. Meanwhile, the various file transport methods and random ports will bring additional problems to the diffusing mechanism.

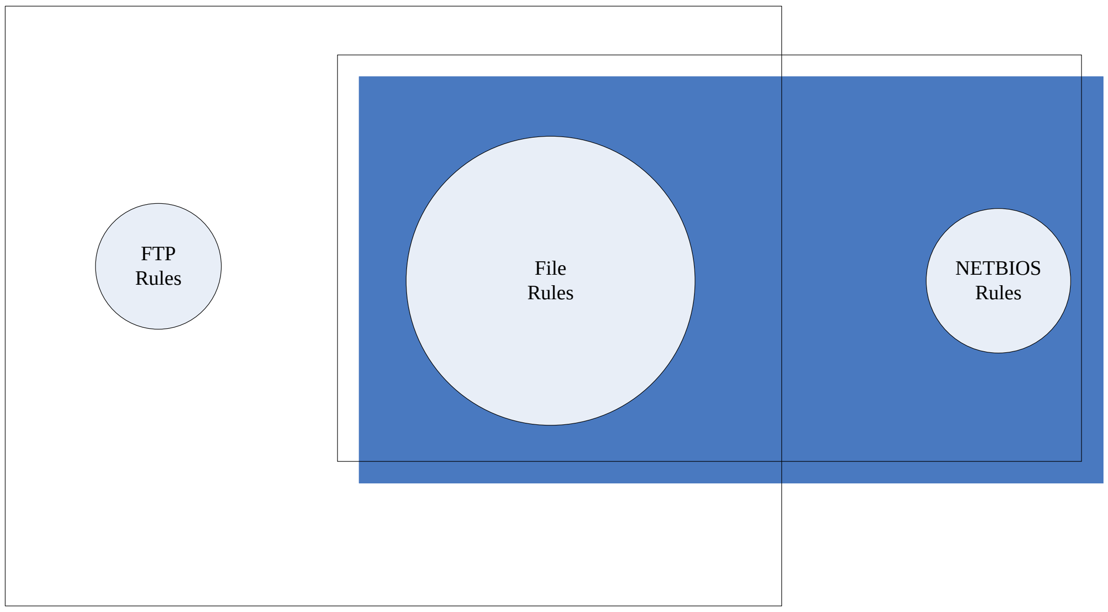


Figure 2: A large number of common characteristic sets bring redundancy

The redundancy of the mechanism is only a superficial problem，in practice, the network virus detection will face more serious problem: the scaling pressure. According to statistics from ASTS#6, up until 2005.7.1, various network worms have a total of 5376 different kinds (table 2). Trojans, backdoors, hacking tools and spyware have added up to 30,000.

classification	**number**
mail worm	2807
instant message worm	172
P2P environment worm	1007
IRC worm	715
other worms	675
Sum	5376

Table 2：2005.7.1 worm classification

Because the virus name and detection rules could be a one-to-many-mapping，the required detection rule sets can be much larger than the virus categories. More than ten thousand rules are needed to detect most of the active viruses, meaning that the linear speed algorithm used by the traditional IDS could become unusable because of the poor efficiency (Table3).

Algorithm Compared				
pattern number	100	200	500	1000
ordinary string match (s)	0. 30	0. 6	1.3	2．5
FA(s)	0.01	0.03	0.07	0.18
AVL engine(s)	0.01	0.01	0.02	0.03

Table 3: experimental data

Referred from: National Key Lab of HIT 2002.9 test report

This table shows the speed of: string matching (averaged), finite automata, and the AVL engine (V1.0). As seen above, string matching has very low efficiency, losing its value when the rule set is large. With regards to the same text length, when the pattern number is less than 100, the time consumed by the AVL engine is almost the same as the automata algorithm. When the pattern number increases, however, the speed of latter falls dramatically compared to the former.

3. VDS Mechanism

3.1Core Architecture

Since the usability of a network virus detection system lies in how to solve the problem of detection with an extra large rule set, we chose a new mechanism which is different from the traditional NIDS. This new mechanism uses the normalization model, with high speed, high granularity matching as its core. It is focused on the matching algorithm, instead of protocol resolution. We call this new mechanism a Virus Detection System (VDS).

In an algorithm design based detection model, we need to divide the network data to be detected into three kinds: one is to be done directly at the binary level, the second one is to be deep processed, and the last one requires a specific algorithm.

Data that needs deep content pre-processing mainly refers to two kinds: one is web pages, where, in order to detect viruses with high granularity, a connection oriented script extractor is needed to detect script viruses in real time. The other kind is encoded mail. Traditional IDS cannot implement detection in the above situations.

So, the normalization method we brought in should regard all the data except that mentioned above as binary level match data. The new normalization model omits accurate protocol resolution. Only the basic data divergence mechanism is reserved. Taking into consideration that some network behavior does not depend on any characteristics and some possibility of reducing false alerts, some characteristics needs related network information. So VDS finally evolves into the mechanism listed in figure 3 below: simple network diffusion, large rule sets, and a secondary mechanism that helps to validate:

- matching rules based on deep content preprocessing
- matching rules for specific algorithms
- binary level matching rules
- binary level matching requiring network information validation
- pure network information rules

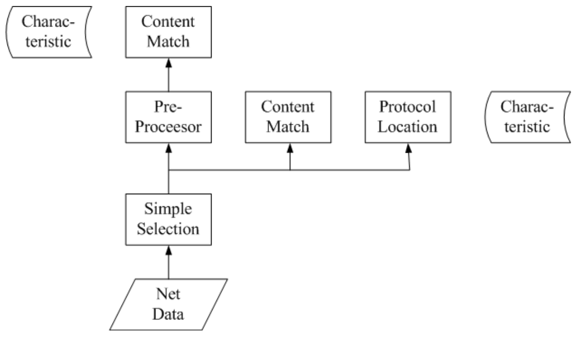


figure3：VDS core mechanism（V.S. figure 1）

3.2 The implementation level of virus detection

Taking into consideration the granularity and detecting efficiency, we offer 3 levels of virus detection in VDS: packet, complete flow and incomplete flow (figure 4). The first two are high speed detection, while the third is low speed detection using a traditional file engine. We discussed that in the xfocus2004 presentation and won’t cover it here.

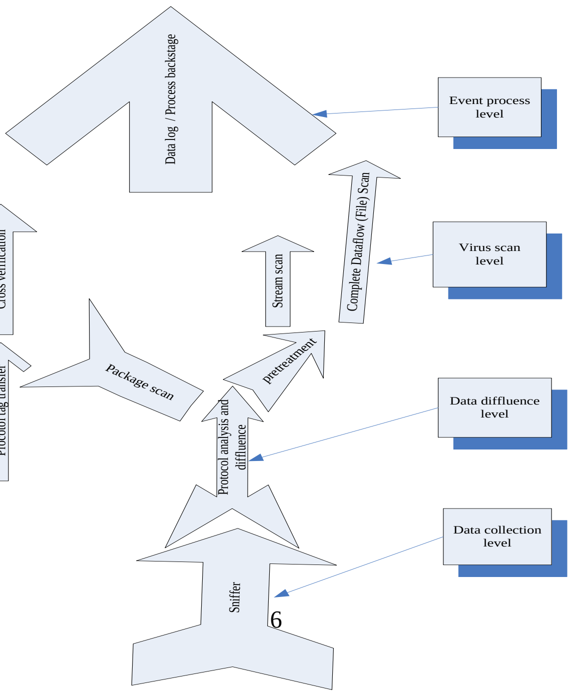


Figure 4: VDS data flow and detection layers

While detecting known viruses, researchers also tried 3 kinds of unknown virus detection: Suspicious script detection based on a NN, power discovery based on short characteristics and behavior deduction.

3.3 Engine Algorithm Optimization

When the number of characteristics increases, the high speed engine based on ordinary records and matching algorithms can also come up with serious problems (figure 5):

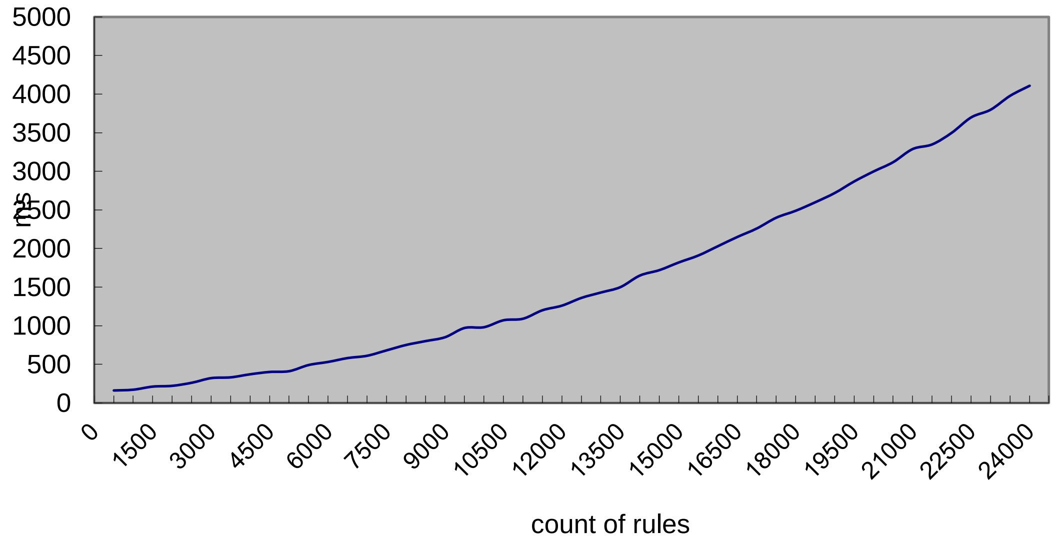


Figure 5: data matching time while records increase (matching object is generated randomly)

This test reflects the performance of the detection engine when detecting the same data, while the records increase by 500 each time. Without specific optimization, when the records are less than 6000, the performance is nearly linearly proportional to the records, but when there are more than 10000 records, the performance drops dramatically, even to the point of being unusable.

Meanwhile, the detection speed also has tight relationship with the input data and the quality of the patterns. Because many viruses share similarities and cannot be distributed randomly, such conditions can also affect the matching speed. Figure 6 illustrates such influence. Obviously, with many detection features, and when using real-world network data to test, the engine will have the poorest performance.

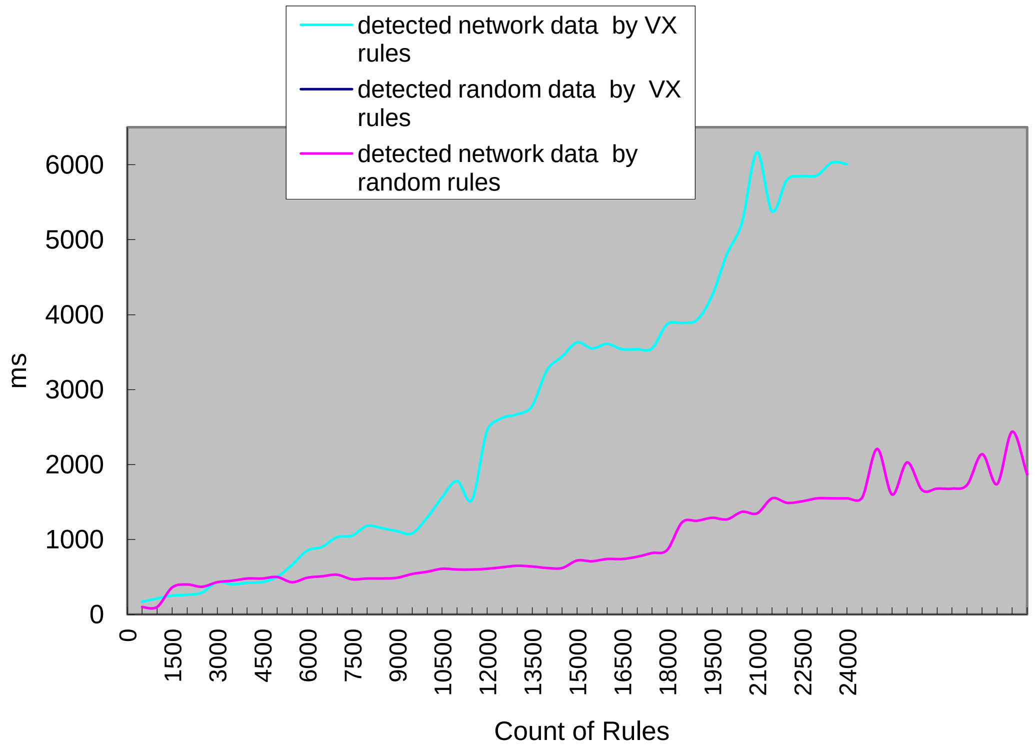


Figure 6: the relationship between detection time and number of rules when the rule set and sample data vary

This algorithm can obviously constrain VDS functions, so it needs to be optimized. Our basic method is to reduce the influence of the similarity among virus signature codes. Figure7 is the efficiency comparison between the engine before and after the optimization.

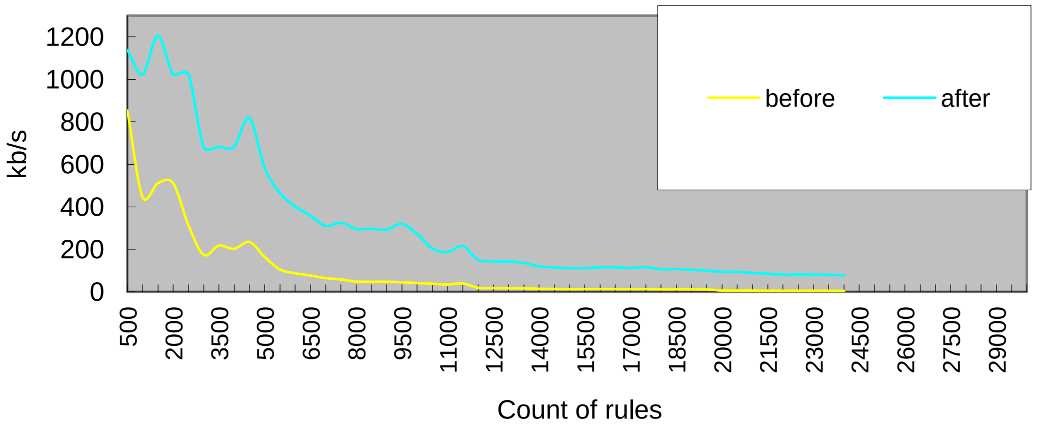


Figure 7: the influence of algorithm optimization on matching speed

3.4 Architecture

The VDS system architecture is showed in Figure 8 below. Sniffer is the network data capturing layer and uses Zero Copy techniques.

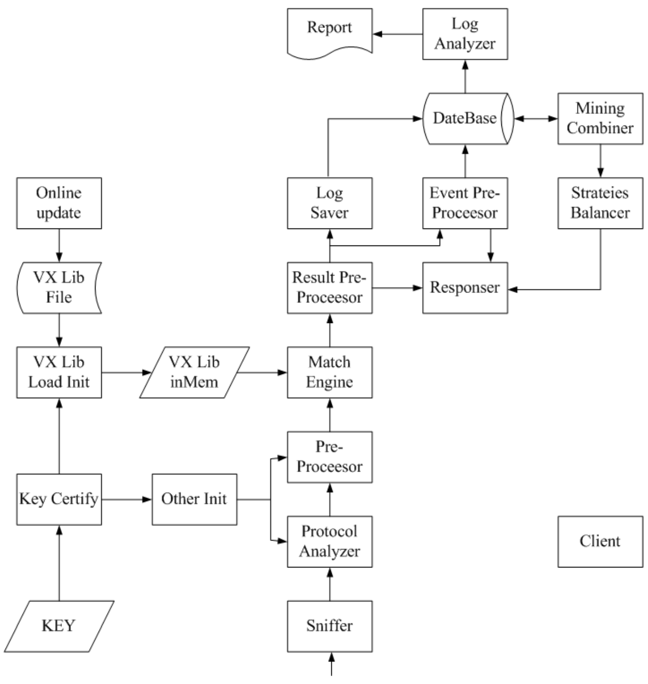
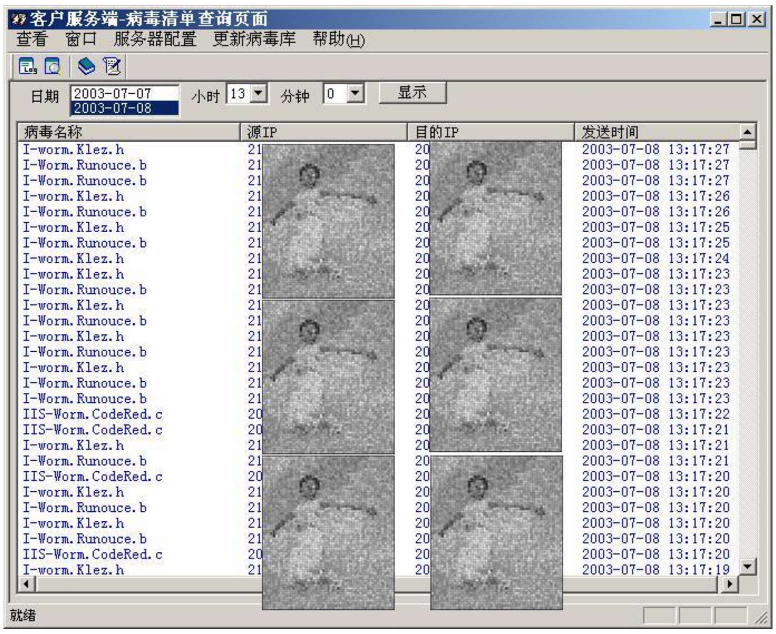


Figure 8 VDS system architecture

3.5 Tests and Experiments

1. The experimental system was installed in December 2002 at the network outlet of HIT with a bandwidth of 1000Mbps

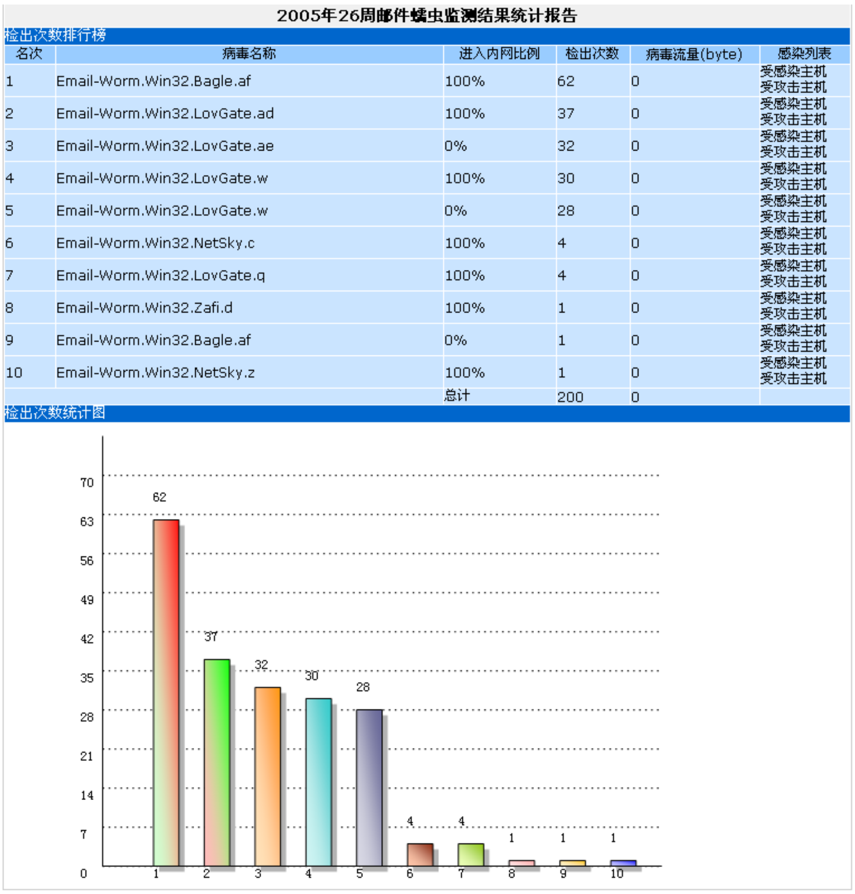

客户服务端-病毒清单查询页面
查看 窗口 服务器配置 更新病毒库 帮助(H)
显示
就绪

日期 2003-07-07 / 2003-07-08　小时 13　分钟 0

病毒名称	源IP	目的IP	发送时间
I-worm.Klez.h	21	20	2003-07-08 13:17:27
I-Worm.Runouce.b	21	20	2003-07-08 13:17:27
I-Worm.Runouce.b	21	20	2003-07-08 13:17:27
I-worm.Klez.h	21	20	2003-07-08 13:17:26
I-Worm.Runouce.b	21	20	2003-07-08 13:17:26
I-worm.Klez.h	21	20	2003-07-08 13:17:25
I-Worm.Runouce.b	21	20	2003-07-08 13:17:25
I-worm.Klez.h	21	20	2003-07-08 13:17:24
I-worm.Klez.h	21	20	2003-07-08 13:17:23
I-Worm.Runouce.b	21	20	2003-07-08 13:17:23
I-worm.Klez.h	21	20	2003-07-08 13:17:23
I-Worm.Runouce.b	21	20	2003-07-08 13:17:23
I-worm.Klez.h	21	20	2003-07-08 13:17:23
I-Worm.Runouce.b	21	20	2003-07-08 13:17:23
I-worm.Klez.h	21	20	2003-07-08 13:17:23
I-Worm.Runouce.b	21	20	2003-07-08 13:17:23
I-Worm.Runouce.b	21	20	2003-07-08 13:17:23
IIS-Worm.CodeRed.c	20	20	2003-07-08 13:17:22
IIS-Worm.CodeRed.c	20	20	2003-07-08 13:17:21
I-worm.Klez.h	21	20	2003-07-08 13:17:21
I-Worm.Runouce.b	21	20	2003-07-08 13:17:21
IIS-Worm.CodeRed.c	20	20	2003-07-08 13:17:20
I-worm.Klez.h	21	20	2003-07-08 13:17:20
I-Worm.Runouce.b	21	20	2003-07-08 13:17:20
I-worm.Klez.h	21	20	2003-07-08 13:17:20
I-Worm.Runouce.b	21	20	2003-07-08 13:17:20
IIS-Worm.CodeRed.c	20	20	2003-07-08 13:17:20
I-worm.Klez.h	21	20	2003-07-08 13:17:19

Figure 9：VDS client (real-time virus event)

2. From March 2003, the experimental system began to issue daily reports to some national departments about the network virus status. The daily report is as follows:



2005年26周邮件蠕虫监测结果统计报告

检出次数排行榜

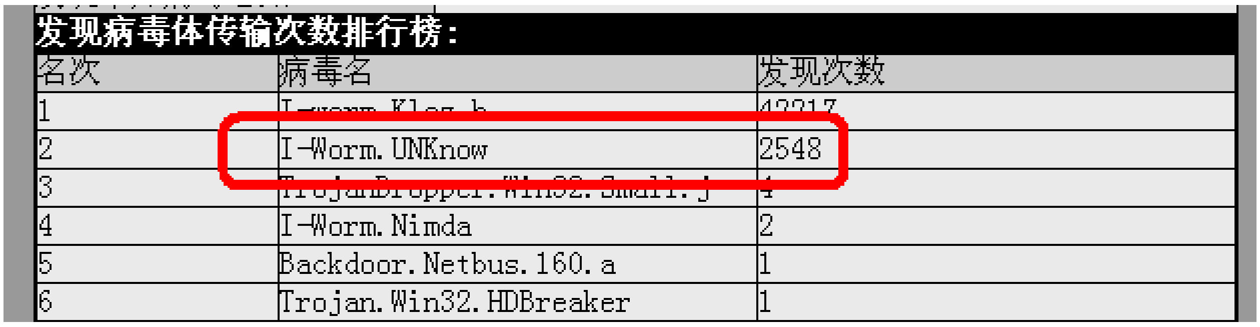
名次	病毒名称	进入内网比例	检出次数	病毒流量(byte)	感染列表
1	Email-Worm.Win32.Bagle.af	100%	62	0	受感染主机 受攻击主机
2	Email-Worm.Win32.LovGate.ad	100%	37	0	受感染主机 受攻击主机
3	Email-Worm.Win32.LovGate.ae	0%	32	0	受感染主机 受攻击主机
4	Email-Worm.Win32.LovGate.w	100%	30	0	受感染主机 受攻击主机
5	Email-Worm.Win32.LovGate.w	0%	28	0	受感染主机 受攻击主机
6	Email-Worm.Win32.NetSky.c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

Figure 10: The email worm report from the 26th week of 2005 (data from network outlet of HIT)

3. Virus Alerts

VDS alerts on several kinds of worms:



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

Figure 11: Data Statistics for Hei Longjiang Education Network Main Nodes (06/05/2003)

On June 5th, we found I-Worm.Unknown was increasing rapidly and on June 6th, it was determined to be I-worm.sobig.f.

4. Data Processing Methods

Because worm events make up a large part of network security events, VDS was faced with a serious record flooding problem. So, VDS adopted a series of data processing methods. In order to describe data processing better, we bring in the DEDL concept:

Definition 3: DEDL (Detection Event Description Language) is a security event normalization method. It uses descriptors to describe the detection events in a unified format and supports general condition deduction.

DEDL defines more than 20 factors such as event type, event ID, Source IP, Target IP, and event time.

4.2 Basic data processing methods used by VDS

In order to provide detection records without flooding the logs, VDS integrates six event processing modes.

a. Inside Technique Combination: Generally refers to internal processing and is transparent to users. Because traditional anti-virus engine use a double code verification mechanism, VDS uses internal combination to avoid repeated records.

b. Parallel Combination: Refers to combining those records with the same source IP, target IP, type and ID into one event. The typical example is mail sending event.
c. Analytical Parallel Combination: Refers to combining those records with different source and target IPs, but the same type and ID. The typical case is a point-to-point scanning event.

d. Radiation Combination: Refers to combining records with the same source IP, but different target IPs. The typical examples are worms and hackers' subnet scanning.

e. Convergence Combination: It refers to combining records with the same target IP but different source IP. The typical examples are DDoS attacks, worm online updates, and IRC connections.
This one is a unique method of VDS.

f. Transportation Chain Combination: Refers to the IP chains formed in the process of infection. We have modified something of this method. Now it is basically based on event name. The source IP of the latter event should be the same as that of the earlier ones.

```
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)]
```

Example 1: the DEDL description of traditional IDS transportation chain（simplified form）

This traditional transportation chain assumes that the sniffer has sufficient coverage, but it can’t guarantee that the formerly infected nodes won’t get infected again.
But the transportation chains of VDS can support infection guessing in inner networks

```
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 belongs to NET(a)
and
time(2)>time(1)
than
Net_Action(Trans,Worm.Win32.Dvldr) [IP(1)-> IP(2) -> IP(3) ->IP(4)]
```

Example 2：VDS transportation chains modified method in DEDL (simplified form)

5. Deep Data Processing and Unknown Malware Discovery

5.1 Behaviour Combination

VDS is based on a large characteristics set, users can detect network worms, but the difficult part is to record the worms’ behavior, otherwise the system cannot interpret whether the network behavior is caused by virus infection.

Through DEDL records and AVML description, we can see the features of signature detection.

DEDL record	AVMLbehavoiur
Net_Action(act)[IP(1),IP(2):445; ;time(1)]	*Virus_act_lib*
Net_Action(act)[IP(1),IP(3):445; ;time(1)]	*Virus*
….	*seek(id=”W02872”;dport=139,445;trans=netbios)*

Net_Action(act)[IP(1),IP(12):445; ;time(1)] *Net_Action(Trans,Worm.Win32.Dvldr)[IP(1)->IP(12);time(1)]*	

Example 3: Worm.Win32.Dvld DEDL event and AVML for Worm.Win32.Dvld and port 445The port 445 scanning event of IP (1) can be combined with the transportation event of Worm.Win32.Dvldr.

5.2 Data Processing Mode

VDS's parallel, radiation and combination process is in place to prevent a flood of event records from hiding the useful records. It is also a good method to detect as-yet-unknown viruses.

Take radiation as an example:

If we assume the event-based topology diagram is as Figure 13, and assume that IRC is sensitive behavior with abnormal convergence status, we can then identify that node A, B, and C are suspicious.

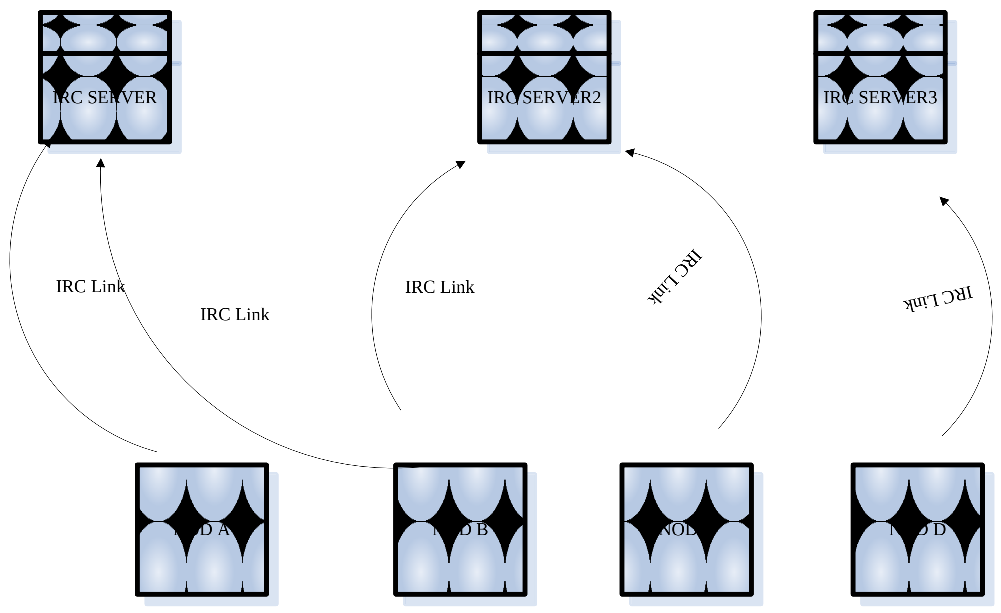


Figure13 network Action of a workgroup

5.3 Discovering Based on Behavior Combination and Data Processing

Because the vulnerable system may be infected by various viruses, simple behavior combination could cover some events caused by other viruses. If we can combine behavior combination and data processing, we can do so as listed in Figure 14. If the behavior is as in Figure 14, then, node B only invalidates the suspicious behavior with IRC_SERVER, while still regarding the behavior with IRC_SERVER2 as suspicious

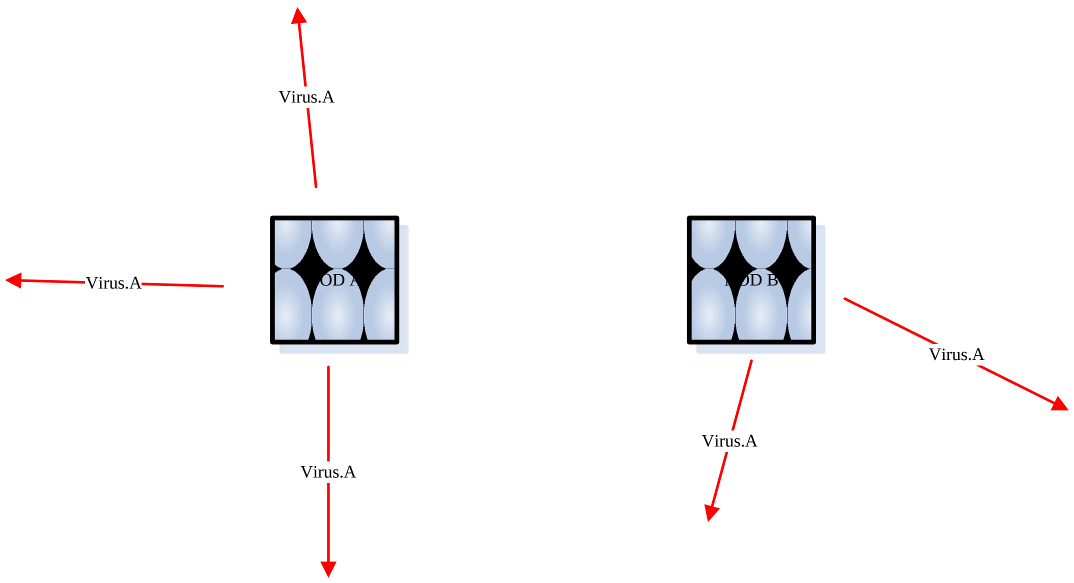


Figure14 virus transportation event and figure 13 combined to analyze

6. The End

A VDS mechanism is a combination of network virus detection research and engineering practice.

VDS has gone one step further than academic network virus research: the traditional academic methods based their research on network behaviors and traffic abnormalities. However, the unknown itself is relative; without an accurate description of the known, the detection of the unknown becomes meaningless. Because VDS brings high-speed accurate detection of a large number of viruses, accurate virus detection on the backbone network become a possibility. Based on this, a suite of research methods came into being.

We’d like to devote this paper to all our hardworking partners on the network virus detection team and all our friends who help and support us.

Reference：

[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*