USENIX Association

Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference

Boston, Massachusetts, USA
June 25–30, 2001

Cost Effective Security for Small Businesses:
A Guide to Open Source Solutions

Sean R. Brown

Applied Geographics, Inc.
Boston, MA

srbrown@appgeo.com

Abstract

As high bandwidth Internet access becomes more readily available at lower cost, many small companies are leveraging the Internet to expand their market share and grow their business. Companies choosing to connect their internal LANs to the Internet often sacrifice the security of local network resources for the sake of expediency and cost. With the availability of inexpensive hardware and the proliferation of open source software projects, highly reliable network security solutions are no longer just for large corporations.

The object of this paper is to provide a general overview of how a small or medium sized business can implement advanced and highly reliable network security solutions using freely distributable and open source software.

1. Introduction

In June of 1999 a destructive worm was released into the wild affecting users of the Microsoft Windows based operating system and the Outlook E−mail client. The worm, known as ExploreZip [1], was one in a series of viruses and worms to exploit known vulnerabilities in Windows network filesharing and Outlook attachment processing [2]. The ExploreZip worm spread rampantly throughout the US leaving many company LANs and personal home computers unusable or severely incapacitated. It destroyed or zeroed out random Microsoft Office documents on user and fileserver hard drives before spreading itself to other susceptible hosts. By some estimates [3] the ExploreZip worm caused up to $7.6 billion damage before slowly fading away. The author of this worm was never caught.

In February of 2000, Yahoo, Amazon.com, Ebay, Etrade, and several other high profile ecommerce sites were targeted by a series of distributed denial of service (DDOS) attacks severely affecting site performance and leaving some sites unreachable [4]. The computer systems used in the attacks were largely owned by individuals, small businesses, universities and large companies, all with dedicated high bandwidth internet connections. Compromised through known vulnerabilities and published exploits the resulting network of infected host systems provided the platform from which the DDOS attacks were launched. In most cases, system owners had no idea that their computers were being used in the attacks.

Recent vulnerabilities in ISC's Bind, Washington University's wu−ftpd, lprng, and the ubiquitous rpc.statd resulted in the spread of three new Linux worms since January 2001 [5][6][7]. These worms perform a root compromise on the vulnerable system allowing the attacker complete control thus providing the launchpad for future attacks. Like the distributed denial of service attacks which struck in February 2000, these compromised systems are capable of forming networks creating the potential for additional DDOS attacks.

The above examples make it abundantly clear that businesses need to take responsibility for protecting themselves and their bottom line from malicious intruders. The rush to get connected was pursued without understanding the potential dangers of being online. Now that they are online, many businesses get buried under the added cost of cleaning up after the latest virus or recovering from a denial of service attack. While the costs associated with these attacks can often be mitigated over the short term, their impact can persist for months and even years as corrupt data, destroyed files and, in some cases, lost business.

While there are many ways to protect networks from external and internal attack, the simple fact is, many small businesses do not have not the financial resources required to implement costly hardware and software security solutions. However, there are numerous freely available security solutions that can, at a minimum, eliminate the upfront software costs of protecting small business networks.

This does not suggest that installing some free software will make a company's network security problems disappear. First, network security is a process which requires a change in mindset about how a business interfaces with the rest of the world. Just as you would not leave the front door to the office open and unlocked when no one is at work, you should likewise, not leave the database server containing all of your companies financial and accounting data connected directly to the internet. Second, any security based software you install, whether it be a proprietary, closed source firewall or a free, open source intrusion detection system, will require configuration, installation and monitoring. Like it or not, the price of network security becomes the cost of doing business online, no matter what the marketing engines say.

The focus of this paper is threefold; first, to look at the various open source or free software available for protecting small networks, second, to describe a case study in open source network security, and third, to summarize the effectiveness of open source firewalling from the perspective of the case study.

2. Tools

While there is an abundance of open source and free software available for securing networks, I will focus on those which I have used in production environments. I will arbitrarily categorize the tools by function in the following way:

- Firewall
- System Integrity/Host Intrusion Detection
- Central Logging
- Encryption Software and Protocols
- Network Intrusion Detection

Obviously, the technological environment in which we live is constantly changing. Software developers are continually updating existing tools and creating new ones to address security threats. While there are a number of tools discussed in this paper, it is by no means, an exhaustive list, nor is that the intent. Rather, this should be considered a primer on some common solutions and a reference for further investigation.

2.1 Firewall

In any networked environment, threats to system integrity come from sources both external and internal to that network. At its most fundamental level a firewall is intended to mitigate external threats by monitoring all traffic entering or leaving a network and allowing or denying that traffic based on packet content. As such a firewall is the first line of defense against external threats.

There are two basic types of firewalls: packet filters [8] and application layer gateways or proxies [9]. A packet filter inspects each packet at the network and transport layers of the Open Systems Interconnection (OSI) model and acts based upon user defined criteria. An advanced form of packet filtering called stateful packet filtering exists when the firewall maintains the state of active sessions. The first packet in the session is compared to the filter rules. If the packet is allowed an entry is created in the state table and successive packets belonging to the same session are allowed without having to pass through the rules test.

```
21/04/2001 22:48:52.238630 STATE:NEW 192.168.2.34,137 −>
    192.168.2.255,137 PR udp
21/04/2001 22:50:51.960067 STATE:EXPIRE 192.168.2.34,137 −>
    192.168.2.255,137 PR udp Pkts 1 Bytes 78
21/04/2001 22:51:18.565050 STATE:NEW 10.10.10.5,1051 −>
    192.168.2.254,22 PR tcp
21/04/2001 22:52:03.960073 STATE:EXPIRE 192.168.10.26,27910
    −> 10.10.10.1,27900 PR udp Pkts 2 Bytes 774
```

The above is an example of state table entries in OpenBSD. The first and second entries are from a NetBIOS name service broadcast which expired after two minutes of inactivity. The third entry is from a newly established SSH session. The fourth is the expiration of a session from a Quake server advertisement.

Packet filtering which does not maintain state compares every packet to the rules list and acts accordingly. The downside of not keeping state is that every packet must traverse the rules before being blocked or allowed. If your ruleset is large, the load on your firewall can be burdensome.

The advantage of having a stateful packet filter is the added control over what is and is not allowed through the firewall. Since IP header information including sequence numbers and time−to−live (TTL) values are kept in state for active sessions, bogus packets claiming to be from an established session will be denied. A stateless packet filter would not have any basis for denying the packet and the traffic would be allowed.

The other basic type of firewall is the application layer gateway or proxy. A true proxying firewall prevents any packet transfer from one side of the firewall to other. In its most secure form, IP forwarding will not be enabled in the kernel. This is known as a dual−homed configuration. Application proxies reside on the firewall and act as surrogates to the original traffic, accepting source packets from one side of the firewall

and creating new packets to forward on to the destination. This design effectively isolates heterogeneous internal systems and the peculiarities of their TCP/IP stacks from having any contact with external devices.

There is a great deal of flexibility in firewall configuration by utilizing elements of both packet filtering and proxying. Large organizations requiring a highly secure environment but also requiring a great deal of functionality may use a stateful packet filter to block all but specific TCP/UDP ports in combination with an application gateway for all traffic from internal hosts. Smaller businesses may choose to only implement non−stateful packet filtering and no proxy services. While the 'best' configuration depends on the needs of the site, the flexibility in firewall options makes it possible to address most requirements.

The design of an inexpensive firewall solution for a small business must account for the operating system and hardware available for the task. The goal is to keep costs at a minimum while not sacrificing security, stability or reliability. Many organizations have rollover plans to replace aging desktop workstations and servers. Older systems can easily be reapportioned to various 'back end' tasks, one of which might be as a network firewall. For organizations lacking this type of replace and reuse capability, there are a number of outlets for obtaining obsolete hardware at a reasonable cost.

While the types of available hardware may vary, the systems that seem most abundant in many businesses are those of the Intel x86 architecture. Other types may be available, such as the Alpha, Sun or Power PC platforms, however, this paper will focus on the use of dated x86 hardware for the given purpose.

There are a number of open source operating systems that have favorable characteristics for building network firewalls. These characteristics include cost, stability, reliability, performance, and scalability. GNU/Linux [10] is probably the most well known UNIX−like operating system running on the x86 system architecture. Its popularity has resulted in not only a large number of available applications, but also a wide range of support options. These include mailing lists, usenet news, web sites and various commercial support offerings.

GNU/Linux is a stable and reliable platform for firewalling. It is easily configured to support different requirements for secure networking such as network address translation (NAT), port forwarding, and packet filtering. The GNU/Linux kernel just went through a recent upgrade to version 2.4. As a result, numerous changes were made to the packet−filtering capabilities requiring a complete upgrade and redesign of existing firewalls if use of the new kernel is desired.

Another open source operating system which may be more favorable than GNU/Linux in many implementations is OpenBSD [11]. OpenBSD is based on the original 4.4BSD public source release. During a two year period starting in 1996, the OpenBSD source code went through an intensive line by line audit for potential vulnerabilities. The default build of OpenBSD has been optimized for security making it an ideal candidate for an open source firewall solution. Since it is based on the original 4.4BSD source tree, it is a very mature OS supporting options like stateful packet filtering, large partitions and filesystem journaling which are only now becoming available for Linux. Performance tests [12] also suggest a faster TCP/IP stack and better I/O than Linux 2.2.x on x86.

Since OpenBSD is optimized for security, it natively supports a number of enhanced encryption capabilities such as SSH remote shells, Kerberos authentication and password encryption using algorithms like blowfish, 3DES, and RSA. OpenBSD also touts the fact that in three years there has never been a remote hole in the default install.

Most UNIX−like operating systems support packet filtering though not all support keeping state. Up until the release of the GNU/Linux kernel 2.4, the kernel did not support stateful packet filtering. However, advances in kernel design and the use of the new netfilter application have made stateful packet filtering possible on the Linux platform. OpenBSD, on the other hand, has provided native support for stateful packet filtering since its inception in 1996.

There are other operating systems which could easily meet the requirements outlined above. Among them include FreeBSD and NetBSD. While, after proper hardening, any of these operating systems would be capable of performing the required tasks, each has been developed with specific functionality in mind, (i.e. FreeBSD: performance on i386, NetBSD: portability to different architectures). Length restrictions require that the discussion of other capable platforms be limited. However, the reader is encouraged to investigate other available open source operating systems and evaluate them according to their own needs.

Normally the default install of any operating system is going to have many vulnerabilities which need to be addressed before the system is placed online. That these vulnerabilities are present in the OS has less to

do with laziness on the part of the manufacturer and more to do with the fact that new vulnerabilities are always being discovered and software is constantly being upgraded.

There are a number of resources useful for maintaining the stability and security of any operating system. OS vendors will usually have information on critical security updates and how to apply patches to secure your system. Among the more prominent resources on newly discovered vulnerabilities is Bugtraq [13]. Subscribing to the Bugtraq mailing list is one important way to get timely information on protecting your site. Two other important outlets for information on currently active threats is the Computer Emergency Response Team (CERT) [14] and the System Administration and Network Security Institute (SANS) [15]. SANS compiles the SANS Top Ten which provides information on the most common threats to computer security and how to eliminate them on your site. The use of these resources provides an important way for security personnel to learn about and ensure the security of their sites.

Table 2.1 − Open Source Operating Systems

OS	*Advantages*	*Disadvantages*
GNU/Linux v2.2.x	Multiple architectures	No kernel support for keeping state (v2.2.x)
	Wide user support	Numerous vulnerabilities in default install.
	Many compatible applications	
OpenBSD v2.8	'Secure by default'	Not as widely supported
	High performance	Steeper learning curve for inexperienced admins
	OS source code thoroughly audited	

There are many open source proxy applications for individual daemon services. Squid [16] is one example of an application proxy for internal http traffic to external web sites. Squid has many advanced features including configurable caching of http traffic, load balancing over multiple proxy servers, and filtering capabilities. For smaller sites with limited bandwidth, Squid can dramatically increase the performance of web browsing through its caching and read−ahead features.

One firewall proxy suite which encompasses many applications in one package and which is available in source form is the Firewall Tool Kit (FWTK) from Trusted Information Systems [17]. The FWTK is a set of applications which act as proxies for production services which would normally be directly connected to the outside world. Daemon services such as SMTP, FTP, HTTP, as well as generic port forwarding are easily handled at the application layer and passed through the firewall by means of a corresponding proxy. FWTK is based on the source code from which the original Gauntlet [18] firewall was built, though their source trees are now widely divergent.

FWTK is not under active development and the license restrictions prevent it from being widely implemented and supported by third party support providers. While the source code is available for free, it is tightly controlled by Network Associates, who purchased Trusted Information Systems, and who now produce the Gauntlet firewall product. The license restricts redistribution of the source code and does not allow third party commercial support of the software. In other words, users may download, compile, and use FWTK within their own organization. However, they cannot pay for, nor can someone sell FWTK support.

There have been many efforts to change these restrictions and make FWTK a true open source initiative, so far without success. However, for the dedicated individual that has the desire to make FWTK work in their environment, the user support community is extremely helpful in solving most, if not all, problems dealing with FWTK implementation. FWTK source will compile on a number of operating platforms including Linux and OpenBSD.

Table 2.2 − Application Proxies

Application	*Advantages*	*Disadvantages*
Squid HTTP proxy	Integrated caching support	Requires extensive tuning to get best performance
	HTTP filtering capabilities	High put a heavy load on a single server configuration with multiple roles
	Load balancing across multiple servers/networks	
FWTK v2.1	High level of security for included application proxies	Restrictive licensing prevents widespread use
	Many user provided enhancements and patches	No longer actively developed by copyright owners
	Strong user support community	

2.2 System Integrity/Host Intrusion Detection

Host based intrusion detection (HID) involves using tools to detect unauthorized modifications to a specific host. Sometimes referred to as system integrity software, HID detects changes in file size, inode number, permissions, etc. which could indicate undesirable activity occurring on the host. System integrity tools also include the capability of storing MD5, CRC32 and/or SHA1 secure hashes of user defined files when the software is initially installed.

Subsequent integrity checks can verify changes to these files.

Tripwire [19] is a well supported, proven HID system. It is available for a number of different platforms and is scalable to the enterprise. It is also a commercial product which can be very expensive for small companies. However, there are two versions of Tripwire which are available as source code. The academic source release is available to qualified educational institutions and academics. It is not the latest version of the software and does not support some of the newer features such as an encrypted database. However, it is a stable and reliable solution if your needs are limited.

Tripwire for Linux v2.2.1 has been released under the GNU Public License (GPL) [20] and is now being actively developed by the open source community [21]. This version supports database encryption which is lacking from the academic source version. Encrypting the system integrity database protects it from an intruder who might otherwise be able to manipulate it to hide their actions. If the database were not encrypted, it would be possible to programmatically modify the system integrity database during a host compromise. Subsequent integrity checks would not reveal the presence of the compromised system files and hide the actions of the intruder. Database encryption is another mechanism by which Tripwire for Linux ensures the integrity of the host.

Since the GPL'd version of Tripwire is written for Linux, it will not compile under other operating environments like OpenBSD. However, precompiled binaries of the Linux based software can be run in Linux compatibility mode on other operating systems like OpenBSD and FreeBSD. This makes the GPL'd Linux version of Tripwire a popular option for host based intrusion detection.

Another HID system is a program called the Advanced Intrusion Detection Engine or AIDE [22]. It is designed to be very similar to Tripwire in its capabilities, however, like the academic source version of Tripwire, it does not yet support database encryption. Its configuration and policy file are very similar to Tripwire making migration a simple task. AIDE has also been released under the GPL and will compile on many different OS's.

Table 2.3 − Host Based Intrusion Detection

Application	***Advantages***	***Disadvantages***
Tripwire v.2.2.1	Centralized integrity checking for multiple hosts	Cumbersome management
	Recently moved from commercial to GPL license	No GUI based central console
	Native DB encryption	Linux only (Plans for FreeBSD and other ports)
AIDE v0.7	Multiple levels of system integrity checks	New product with few bells and whistles
	Intuitive configuration file	No encryption
	Multi platform support	No site management

2.3 Central Logging

System logs provide a critical layer in monitoring the security of your network. Because of this, it is often the first thing to be tampered with by a malicious intruder trying to cover their tracks. Syslog provides an easy way to send system log messages to a remote server dedicated as a central log repository. By having a replica of all system logs for a given machine you are able to protect yourself from having the original logs modified.

A convenient way to implement centralized logging is by using the remote logging capabilities of syslog. A central log server listens for log messages sent from remote servers. The messages are sent to the central server in plain ascii text using UDP as a transport. However, sending system logs over an untrusted network in plain text makes it very easy for anyone with a packet sniffer to see those logs without having to gain access to either the logging host or the remote server.

One way to alleviate this issue is to send the logs over an encrypted channel using a combination of syslog−ng [23] and stunnel [24]. Stunnel allows you to create an encrypted tunnel between two machines making it possible to send sensitive data over the network in a secure manner. Stunnel uses SSL to build a secure TCP connection on arbitrary ports between two devices. Unfortunately, since syslog uses UDP as a transport it is prevented from functioning with stunnel.

Syslog−ng functions as a replacement for syslog on most UNIX−like operating environments. It has many new features like regular expression matching of system log messages, hashing of log files, and forwarding of logs over TCP connections. Using syslog−ng in combination with stunnel, it is possible to send your logs over an encrypted tunnel to your central log server.

Monitoring system logs is not the most exciting task, especially if you have a central log server collecting logs for many machines on a network. It can be very tedious without an easy and effective way to parse the

Table 2.4 − Centralized Logging

Application	*Advantages*	*Disadvantages*
Syslog	Fast delivery of log messages	UDP protocol means unreliable delivery of log messages
	Standard UNIX software	
Syslog−ng	Uses TCP for reliable log delivery	TCP has added overhead with possible load issues
	Encryption capable using third party TCP tunneling	
	Pattern matching logging	

Table 2.5 − Logfile Monitoring

Application	*Advantages*	*Disadvantages*
Logwatch v.1.1.1	Standard pattern matching syntax	Periodic vs. realtime monitoring
	Easily configured for multiple log file formats	No active alerting capability
Swatch	Realtime monitoring vs. periodic checking	Cannot monitor multiple logs in different locations
	Active alerting and program execution	

data you want. Luckily, there are a number of tools available to help.

Psionic Logcheck [25] is a package which monitors system log files at regular intervals searching for user specified regular expressions and sending alerts if any of those expressions are matched. The software has a compiled executable which monitors log file changes between specified run intervals, and a shell script which is configured with variables pointing to the log files to be monitored. There are also a few configuration files listing the regular expressions to match against. The regular expressions use a standard syntax and are easily modified for different environments.

Another tool for monitoring log files is a package called Swatch [26] or the Simple Watcher. Swatch has been around for many years and was first introduced at the 1993 LISA conference. It performs some of the functions of Logcheck but is a smaller package and has some additional functionality. Swatch is run once and monitors your logs in real time versus running as a scheduled cron task. This has many advantages in situations where real time notification of security events is absolutely necessary.

Some of Swatch's other capabilities include executing arbitrary code, color coding program output, and piping output to commands, all based on user defined criteria. However, Swatch is limited to monitoring a single logfile at a time, unless multiple instances are run concurrently. This makes it difficult to configure monitoring multiple logs under one process.

2.4 Encryption Protocols

For many years, all network traffic was unencrypted, plain ascii text, easily viewed by anyone running a packet sniffer. Even today, most email, ftp downloads, irc, and web traffic transport their data in plain ascii text. While encryption technology is gradually making its way into mainstream, everyday use, there are definitely programs available now to allow the use of high encryption for most, if not all, traffic over untrusted networks.

Encryption of network traffic normally occurs in one of two ways. A person uses an application to manually encrypt the data prior to sending it over the network. This is the common method for encrypting messages like email or news posts. The encryption is handled by the user at the application layer. Only the specific data encrypted by the end user is secured. Mail headers and any other information outside the envelope of encryption is left in plain text.

Common programs for encrypting files and messages include Pretty Good Privacy [27] and GNU Privacy Guard [28], both of which use a technology called public key cryptography. It is far beyond the scope of this paper to detail the technical aspects of public key cryptography, suffice to say that this technology is making military grade encryption possible for those wishing to protect their files and data.

Table 2.6 − Encryption

Application	*Advantages*	*Disadvantages*
Pretty Good Privacy, PGP	The first widely accessible encryption program	Most development in recent years has been on the Windows client
	Support for multiple algorithms	Now a commercial program with free version available
GNU Privacy Guard	Open Source	Clumsy GUI support
	Backwards compatible with PGP secret keys	
	Wide user support and GUI add−ons	

Another method of encrypting network traffic is by either using an application which supports native transport of encrypted packets such as OpenSSH [29], and OpenSSL [30] or by sending all data and packets through an encrypted tunnel using an application like Stunnel or a protocol like IPSEC [31] and Point−to−Point Tunneling Protocol (PPTP) [32]. Encrypted tunnels are the basis for VPN's. For applications that do not natively support encryption, a tunneling

protocol is an ideal way to encrypt all the traffic between systems on opposite ends of a tunnel.

Table 2.7 − Tunneling Applications/Protocols

Protocol/App	***Advantages***	***Disadvantages***
Stunnel	Quick tunnel for specific applications	Works on the transport layer and limited to TCP
IPSEC	Functions on the network layer	Limited application support
	Highly secure	
	Internet standard	
PPTP	Application support on multiple platforms	Known vulnerabilities
		Not an internet standard
		Becoming obsolete with advances in IPSEC support
OpenSSH	Terminal access	No UDP encryption
	Capable of tunneling arbitrary connections	
	Useful for encrypted file transfer	
OpenSSL	Create self signed SSL certificates	Self signed certificates are not trusted by remote users
	Open Source alternative to Verisign	Need a root Certificate Authority for production sites

2.5 Network Intrusion Detection

While host based intrusion detection monitors the integrity of individual hosts, this does nothing to detect or monitor malicious traffic on the network. Network based intrusion detection (NID) fills this gap by providing a way to see exactly what packets are traversing your network allowing you to evaluate whether or not they should be permitted. There are two types of NIDS: anomaly detection and pattern matching [33].

Anomaly detection compares network traffic patterns to an established baseline of normal activity. Deviations from that norm are flagged for further analysis. One benefit of anomaly detection is that it does not rely on predefined signatures for detecting abnormal activity. Rather it uses a statistical analysis to determine whether current traffic falls within the parameters of 'normal' activity. In this way it can detect previously unrecognized activity or new intrusion methods which are not detectable through other means.

Pattern matching, as the name suggests, compares packets against predefined signatures. The goal is to detect specific traffic based on a given signature and then optionally taking some type of action. Signatures are established through analysis of previous intrusions. Characteristics which can be used to identify a specific intrusion attempt, such as a particular combination of TCP flags or a TTL value which falls within a certain range, can be used to create a profile of the activity making it detectable through packet analysis. The advantage of pattern matching is that it can be a quick, efficient way to screen for known intrusion activity. However, unlike anomaly detection, it cannot detect unknown attacks.

NID systems are normally either installed directly on a firewall or on a separate system configured to listen on all traffic going through the firewall. Most modern switches provide this capability through port mirroring.

There are a number of freely available NID systems. One popular and highly effective NIDS is Snort [34]. Snort, released under the GNU Public License, is under constant development and is continuously having new capabilities added. Snort relies on user defined rulesets to detect and flag suspect network traffic. The rules language is very flexible allowing pattern matching on packet headers and data content. These sample rules [35] look for a specific virus and backdoor activity.

```
alert tcp any any −> $HOME_NET 139 (msg:"Virus − Possible QAZ
   Worm Infection"; flags:A; content: "/71 61 7a 77 73 78 2e 68 73
   71/"; reference:MCAFEE,98775;)
alert tcp $EXTERNAL_NET 27374 −> $HOME_NET any (msg:
   "BACKDOOR SIG − SubSseven 22"; flags: A+; content:
   "|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485;)
```

Newer plugins for Snort enable more advanced features like statistical anomaly detection and fragmentation reassembly. There are a number of logging options a user can choose such as logging directly to syslog, a flat ascii text file, XML, or a number of client server databases including MySQL and Postgres SQL.

Snort was designed to be easily extended through plugins which add new capability to the core program. There has also been a minor explosion in Snort add−ons and partner programs. Some of these work directly with Snort output to create customized reports, correlate attacks, and provide active, real time alerting of intrusion attempts and suspicious network traffic. Most of these add−ons are also GPL'd open source and are freely available.

A support tool useful for analyzing data captured by Snort is the Analysis Console for Intrusion Databases (ACID) [36]. ACID is a web application which interacts with the database tables populated by Snort. ACID has features which allow for close examination

of individual packets and the calculation of summary statistics, useful for observing trends and patterns. Whois lookup capabilities and detailed querying of the packet database make it possible to thoroughly investigate where a packet originated, what it contains, and its context with other network traffic, whether concurrent or not. While there are other ways to extract usable information from Snort logs, the combination of ACID and MySQL is a very convenient and useful option (Figure 2.1).

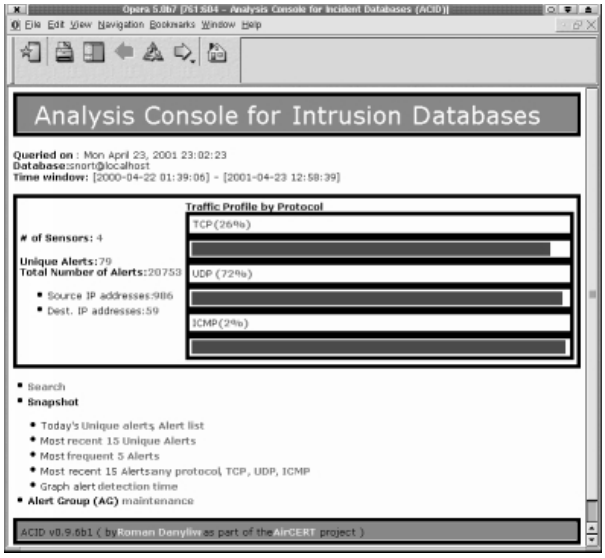

Figure 2.1: ACID Console

Another pseudo NID software is tcpdump [37], included on most UNIX−like OS's. More of a highly configurable packet logger, tcpdump is based on the same packet capture library as Snort and therefore suffers some of the same limitations under heavy load. Tcpdump is a great diagnostic tool but was never intended to perform advanced intrusion detection. However, it is included here as part of an overall combination of applications useful for intrusion detection.

Ethereal [38] is a GUI based protocol analyzer which, like Snort and tcpdump, also uses the pcap packet capture library. Ethereal, while capable of capturing raw traffic, really shines in its usefulness in analyzing tcpdump format packet capture files. Raw packets are shown in both hex and ascii and deciphered into an easily readable format for easy scanning (Figure 2.2).

3. Case Study − Applied Geographics

While it is one thing to describe the free tools available for helping to secure a network it is often helpful to provide an example of them implemented in a real world situation. A case study can be very informative regarding both the problems encountered,

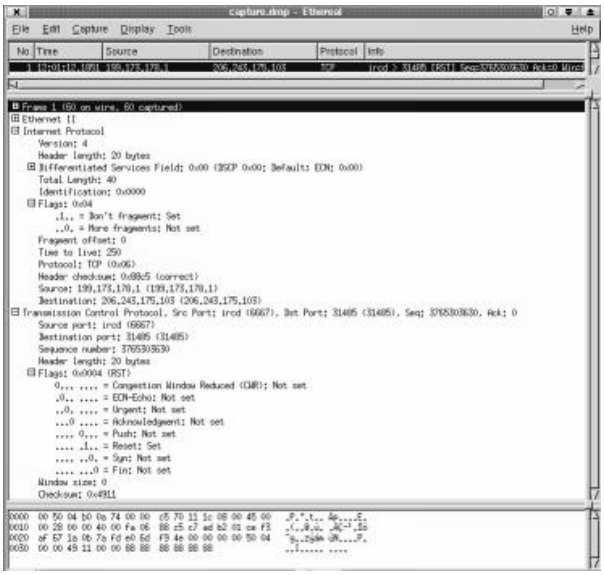

Figure 2.2: Ethereal Console

as well as the benefits, should someone want to replicate the environment for themselves.

Applied Geographics, Inc. (AGI) [39] is a Geographic Information Systems (GIS) consulting company specializing in municipal GIS, web based map server development, and facilities database integration for large organizations. Much of the companies web development projects evolve as prototype map applications running on hosted Microsoft IIS web servers at AGI. These web based applications are made available to clients during the development phase. Since client access is very mobile, it is not possible to restrict access through IP address filtering. Therefore, diligent efforts must be made to protect both AGI's and our clients interests.

Table 2.8 − Network Intrusion Detection

Applications	***Advantages***	***Disadvantages***
Snort	Very flexible detection and reporting	Can drop packets under load
	Multiple plugins available	
Tcpdump	Advanced packet logging	Can drop packets under load
Ethereal	GUI interface	Known root exploits
	Excellent interface for analyzing packet logs	

3.1 Past Suffering

In June of 1999, AGI suffered the results of the ExploreZip worm which swept through large segments of the Internet [40]. The worm tore through the companies fileservers zeroing out thousands of documents and development project files and generally wreaking havoc on the email system. This

was not an uncommon occurrence for companies relying on Microsoft technology for their email and office productivity software.

If anything positive came out of the ExploreZip event, it was a greater awareness of the threats that exist to companies directly connected to the Internet. When I came to AGI in September of 1999, ExploreZip was still fresh in the minds of the company. It was made clear that protection from the type of threat ExploreZip represented, i.e. opportunistic exploitation of critical system wide vulnerabilities, was a top priority.

I conducted an audit of our existing exposure and found a number of things which needed attention. These included an unpatched Linux 2.0.x firewall with several vulnerable daemons including bind, wu−ftpd, and rpc, no packet filtering, an ftp server open to anonymous uploads, an open mail relay which I later found was being used as a spam relay, telnet access, and intranet fileserver browsing access available from outside the firewall. Any one of these vulnerable points of exposure could have resulted in disaster and it's a wonder that the phone wasn't ringing off the hook with angry system administrators from remote sites.

3.2 Needs Assessment

The first step in designing a security solution is to determine what exactly needs to be secured. This may sound absurd but having a set of guidelines to address specific network security issues is very important. What is the required functionality for internal users? What external access is required to internal resources? Is it necessary to encrypt all network traffic or only external authentication traffic? These are questions which need to be answered before proceeding to installation and configuration of particular software.

At the time of the initial assessment the network environment at AGI was represented by a 40−50 node TCP/IP network running a heterogeneous mixture of Windows NT Server, Windows NT Workstation, Linux, and Digital UNIX. Connectivity to the Internet was a through a 416 Kbps SDSL link. Since we were actively replacing our older workstations with newer systems, we had a potentially ready supply of backend, special purpose servers at our disposal.

Required functionality was defined in terms of what services to provide employees on the inside, and what services were required to serve AGI's clients on the outside. The design of the solution also needed to be flexible enough to support future added functionality without sacrificing security or core services. As always, cost was an important element in the design decisions.

3.3 Internal Services

AGI's network was a relatively open environment with few restrictions on access to resources. As a small company, employees were used to having mostly unfettered access to any and all resources within the corporate network. Therefore, most efforts to protect the internal network were focused on external threats. To minimize potential threats posed by accessing remote resources, access to common internet resources needed to be proxied. Also, network address translation was required for any connections where a proxy was not available.

To accommodate these requirements a new firewall was designed using Red Hat Linux v6.1 [41] with a 2.2.x kernel. While the GNU/Linux kernel v2.2.x does not support stateful packet inspection, it was felt at the time that the trade off of using Linux for its software compatibility versus an OS platform that maintains state was acceptable. Red Hat was readily available and therefore chosen as the firewall OS.

While a default install of most operating systems is quite insecure, it is fairly straightforward to put the system into a secure state through post installation modifications. A spare 100MHz Pentium system with 96MB of RAM was chosen as the hardware platform for the new firewall. A minimal install of Red Hat 6.1 was then hardened by removing all unneeded services, locking down user authentication, installing OpenSSH for terminal access, and creating ACL's for access to both local and remote services.

Since the main concern was for external security there was no requirement to encrypt the transmission of syslog messages to the central server. A central log server was installed and configured to use syslog as a listener for the internal network. Syslog was also a requirement due to the heterogeneous nature of AGI's network. Syslog−ng is not supported on older Digital UNIX and Windows NT systems, all of which needed to forward their log messages to the central server.

Since the amount of traffic passing through the firewall was relatively minimal the packet filter and application gateway were configured on the same device. In larger implementations, this may not be appropriate.

Squid was installed to provide gateway services to all external http resources. The added functionality and performance of Squid made it an appropriate choice for AGI. Since http traffic is the primary external

resource accessed at AGI, it was felt to be important to have a proxy in place. Many other services, including Real Audio and ICQ, have the ability to use http as a transport obviating the need for additional proxy support beyond Squid. Other resources, such as external ftp servers, were kept unproxied.

Network address translation (NAT) hides internal devices that access external resources by replacing the source ip address of the internal system with the ip address of the firewall. The ipchains [42] utility allows for the easy configuration of NAT using one simple rule:

```
# ipchains −A forward −i $EXT −j MASQ
```

This will masquerade all packets exiting the local network on the external interface.

3.4 External Services

Providing access to services and resources from external, untrusted networks requires careful planning to minimize threat exposure. It was necessary for AGI to provide a number of services to clients, employees, and the general public. These services consisted of an SMTP mail server for local and remote mail delivery, web hosting for both AGI and our client prototype applications, an ftp server for anonymous downloads and authenticated uploads, a VPN for telecommuting, and an administrative console capability.

The following action items were established to satisfy the functional requirements of the AGI network. First, a packet filter was needed to provide access controls to the AGI network. These included egress filtering of RFC 1597 private addresses. While this step does not necessarily protect AGI, it does promote responsible network management. Second, daemon services should be proxied where possible. Third, AGI employees should be able to access internal network resources from home or on the road. Fourth, at a minimum, all authentication of external sessions to internal resources (web access, VPN, terminal) should be encrypted. If possible, the entire session should also be encrypted.

The v2.2.x Linux kernel uses a program called ipchains to manage packet filtering. The three default chains, input, output, and forward, were all defaulted to deny access. By denying everything and then selectively allowing specific traffic, you create an awareness of the nature of all traffic entering or leaving the protected site. You also protect yourself from overlooking a type of traffic that would normally not be allowed. At AGI, it was necessary to allow access to a number of specific TCP and UDP ports including 21 (FTP), 22 (SSH), 25 (SMTP), 80 (HTTP), 143 (IMAP), 443 (HTTPS) and 993 (IMAPS). Servers listening on these ports were regularly monitored and patched to prevent possible compromise.

Egress filtering is a method to prevent broken packets or packets with non−routable addresses from being routed outside the protected network. All packets with RFC 1918 source addresses (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16) should be blocked from routing to the internet. This is good networking practice and reduces the routing of unwanted traffic.

The TIS Firewall Toolkit (FWTK) provided the application gateway support required for SMTP transport. The original source code for FWTK's smap proxy does not support a restricted mail relay or spam filtering. The end result is that while sites are protected from exposing their mail servers directly to the internet, the proxy, itself, is vulnerable as an open spam relay. However, there are a number of highly effective patches to the original FWTK source code which add the necessary open relay controls while also providing support for spam filtering. This makes the FWTK smap proxy a usable mail proxy for AGI's site. The FWTK also supports proxies for a number of other application services such as FTP, HTTP, X server, Telnet, and SMTP. However, the SMTP gateway was the only FWTK server proxy used at AGI.

In addition to filtering and proxying network traffic to and from AGI, it was necessary to encrypt as much of the remaining traffic as possible, protecting usernames, passwords and sensitive data. E−mail encryption is still fairly non−standard and cumbersome making it difficult to implement in any mildly heterogeneous environment, let alone with any outside clients and contacts. These unresolved issues made it exceedingly difficult to control the use of application level encryption. However, it was possible to control the encryption of user logon sessions to IMAP mailboxes, access to internal web based information, and remote access.

The primary concern was authenticated, encrypted access by AGI employee's and clients to protected resources. Creating a root Certificate Authority (CA) for AGI permitted the creation of self−signed web and email server SSL certificates. These certificates would permit SSL access for testing and pre−production phases of the development cycle. OpenSSL was used to created a root certificate which was then used to sign certificates created by web and email servers. Most modern web and email servers including IIS, Apache and Exchange, are capable of

utilizing SSL transport of data. Once the certificates were installed on the servers, users of the site needed to install the CA root certificate as a trusted root authority in the web browser. This prevented errors about untrusted root certificates from being generated by the browser. Email clients were configured to use port 993 (IMAP over SSL) to connect to the IMAP server. These two steps eliminated 90% of the plaintext authentication and data stream to/from AGI's protected resources.

One of the core requirements for a new firewall solution was an effective remote access capability that did not open AGI to excessive risk. Any solution needed to rely on encryption of initial session negotiation, authentication, and data flow.

As a GIS solutions provider, there are specific software dependancies that AGI must support. In the past this software was multi−platform allowing the use of any number of operating system and hardware choices. AGI's legacy GIS applications run on aging Digital UNIX servers accessed through X terminal sessions. These applications and hardware platforms continue to be supported at AGI. However, changes in the GIS software market have dictated a migration to a Microsoft−centric operating platform at the desktop. In addition, widely available broadband internet connections have made telecommuting a viable alternative to battling the morning commute for many of AGI employee's. Initial research into a solution enumerated a couple of potential open source solutions such as PoPToP [43] and FreeS/WAN [44]. However, both were fairly new initiatives which did not fully support the type of access required, vis à vis full replication of the desktop environment at AGI through an encrypted tunnel. The decision was made to use PPTP on NT Server passing the traffic through the firewall.

Since the focus of this paper is on the use of free, open source security solutions, a discussion of the relative merits and limitations of using Microsoft's PPTP implementation versus some of the more reliable, secure or open source VPN solutions on the market will be avoided. Given further refinement of the existing initiatives, an open source solution would provide an attractive alternative. However, while most open source VPN solutions focus on providing secure router <−> router links using IPSEC thereby obviating the need for dedicated client software, any VPN remote access solution for workstation <−> server would require a software client developed for the Windows desktop. Commercial products are available which fill this gap, however, open source VPN client software that supports IPSec and is able to run on the Windows desktop is noticeably absent.

In addition to a secure remote access solution for AGI employees, a secure method of administering the network while off site was also required. OpenSSH provided a highly reliable and secure method for establishing terminal connections to AGI network servers from remote sites. OpenSSH has many configurable parameters including key length, cipher, and authentication mechanisms such as Kerberos, RSA key authentication and username/password combinations. OpenSSH also provides methods for creating encrypted tunnels over arbitrary TCP ports, secure file transmissions, and secure X window negotiation if desired.

3.5 System Integrity

One of the key problems with host compromises is that they are usually not discovered until long after the compromise took place and the intruder has already damaged your site or someone else's. Recent attacks on exploitable vulnerabilities in bind, rpc, wu−ftpd, and lpd make it abundantly clear that undiscovered system compromises contribute to the spread of worms, both kiddies and automated, looking for further victims. Unchecked, these compromised systems pose a serious risk to the stability of the internet.

A good host level intrusion detection capability is a necessary component of any system exposed to direct connections from the internet. At AGI, it was necessary to provide host based intrusion detection on the GNU/Linux firewall as well as any other Linux systems installed on the network. Since the goal was to use open source software, system integrity software for other OS platforms will not be presented.

Two open source solutions were previously described. While both Tripwire and the Advanced Intrusion Detection Engine (AIDE) provide system integrity checking, only the recently open sourced Tripwire 2.2.x for GNU/Linux has built−in capability of encrypting the system database. This functionality is planned for AIDE, however, with the GPL'ing of the Linux client, the maturity of the application, and with the support for running Linux binaries on BSD systems in compatibility mode, Tripwire provided an ideal solution at AGI.

3.6 Network Intrusion Detection

Solutions for an adequate firewall and host level intrusion detection provide a solid base for protecting private networks. However, these tools do not actually allow you to see the traffic entering and leaving the network. A network based intrusion detection system

provides a means to identify unwanted network traffic and take appropriate action.

AGI required a system which could be modified to detect new and customized signatures, log alerts to a central database, and provide the means to analyze detects. Snort provided the ability to create advanced rulesets for detecting unwanted network activity, and report that activity to a remote database. The Snort configuration at AGI combines a subset of rules made available by the Snort development team and a custom ruleset created to account for known traffic patterns on AGI's network. Alerts are logged to a local binary tcpdump format logfile in addition to a remote MySQL database. Analysis of all alerts logged to the MySQL database is conducted using ACID.

In addition to the use of ACID for analyzing the Snort alert database, Ethereal's filtering and sorting capabilities make it an extremely useful companion for analyzing the tcpdump format binary files created by Snort. The drill down capability of Ethereal allows for a complete and precise view of each packet.

The use of these tools highlights another important benefit to using open source software for network security. There are no limitations to the combinations of software that can be used for any specific objective. Commercial software gets expensive if you have to keep buying new packages for additional functionality. If you don't like an open source application, there's no financial incentive to try and 'make' it work if it simply will not serve your purpose. If an application has some useful features but does not provide all the functionality required, there are other packages available which can either replace the existing software or compliment it with the new features. Either way, the investment in the software costs you nothing.

3.7 Cost Considerations

As stated in the introduction, the cost of network security is part and parcel of the cost of doing business online. Hardware, software, time and skills, all come at a price. Formal training and certification programs can be expensive. Conflicting priorities and responsibilities can make it difficult for system administrators to spend the time needed to learn new skills and keep current on the technology. Fortunately, with the availability of inexpensive legacy hardware, free and open source software, and a wealth of information online, those costs have neither to be exorbitant nor unmanageable. In fact, the money spent on network security now may be an investment in your businesses online future.

The choices made at AGI on software and hardware paid off immensely. Through the use of spare hardware, a solid open source OS, and the open source security tools described above, it was possible to implement a highly secure network environment at AGI. The software required to build the solution cost nothing. The cost incurred in implementing the solution was the time spent initially building the system, and the subsequent hours spent updating, monitoring, and maintaining the software. Since these duties fall within the role of System Administrator at AGI, the costs were negligible as a separate line item and were well justified.

3.8 Future Directions

In the year and a half since the original solution was implemented, the GNU/Linux box running on a five year old Pentium 100 desktop workstation, has never crashed, failed or been compromised. It was rebooted twice, due to physical location changes and handled all the network traffic for 35 employees, six hosted production web sites, 6 staging and developmental beta web sites, and up to 10 concurrent VPN connections without any interruption of service.

Recent changes in the network infrastructure at AGI, demands for additional bandwidth, and additional hosting requirements, required moving the firewall to a larger box and adding stateful packet filtering capabilities. Since stateful packet inspection is a recent addition to GNU/Linux and may have some lingering unresolved vulnerabilities [45], and since software compatibility issues are no longer an issue in using *BSD based systems, OpenBSD was used rather than Linux. The move will allow AGI to maintain existing capabilities, while enabling stateful packet filtering for better performance and less overhead.

The bandwidth at AGI has moved from 416Kbps SDSL to two full 1.544Mbps T1's. The network performance statistics on *BSD vs. Linux [12] as well as the reputation of OpenBSD with its focus on security made the decision to switch an easy one. The capabilities and reliability of open source operating systems and software make them an extremely attractive alternative to commercial OS's and security software.

4. Summary

In the mid−1990's, when computer programmers and system integrators realized that a date change from the year 1999 to 2000 might have major implications for government and business computer systems, billions of dollars were spent resolving the problem. After literally millions of person hours spent by developers

and technicians repairing code and fixing systems Y2K fizzled on New Years, 2000. Some argued that the money was ill spent since nothing happened. Their logic does not account for the diligent efforts of those who worked on Y2K's demise. It rather suggests that "nothing was going to happen anyway so why did we spend all the money?"

Network security suffers from similar misconceptions. If you take steps to minimize your exposure and suffer no intrusions you are doing your job. But then was all the money and effort really worth the expense? One need only refer to CERT, SANS, NIPC, or a number of other resources specifically designed to keep users informed on the threats that exist.

As mentioned above, the solution implemented at Applied Geographics has never crashed, failed, or been compromised. However, this is not to say that people haven't tried. Since April, 2000, the monitoring of AGI's network resulted in over 30,000 events, including numerous worm detections, root exploit attempts, errant packets and useful traffic analysis leading to network design changes and improvements in overall system performance.

While cost is always an important factor in deciding how best to secure a network, this paper has attempted to show that one need not spend thousands of dollars on commercial software. Any network security solution is going to cost money in the form of someone to provide support, installation and monitoring. However, to believe that you must also incur the added cost of the software is to overlook the solid, cutting edge, and secure open source solutions that are available for free.

5. Acknowledgements

I would like to thank Ted Faber and Peter Girard for their extremely helpful comments and suggestions. Any errors in content are, of course, solely the responsibility of the author.

6. References

[1] Computer Emergency Response Team. CERT Advisory CA−1999−06. http://www.cert.org/advisories/CA−1999−06.html, June, 1999.

[2] Computer Emergency Response Team. CERT Advisory CA−1999−04. http://www.cert.org/advisories/CA−1999−04.html, April, 1999.

[3] Ohlson, K. "Viruses, Other Attacks Cost Businesses $7.6B: Report". http://www.computerworld.com/cwi/story/0,1199,NAV47_STO28244,00.html, June, 1999

[4] National Infrastructure Protection Center. Alert 00−0034. http://www.nipc.gov/warnings/alerts/2000/00−034.htm, February, 2000.

[5] Vision, M. Ramen Internet Worm Analysis. http://projet7.tuxfamily.org/docs/security/ramen.html, 2001.

[6] System Administration, Networking and Security Institute (SANS) Global Incident Analysis Center (GIAC). Lion Worm. http://www.sans.org/y2k/lion.htm, March, 2001.

[7] SANS GIAC. Adore Worm. http://www.sans.org/y2k/adore.htm, April, 2001.

[8] Strom, D. "The Packet Filter: A Basic Network Security Tool". http://www.sans.org/infosecFAQ/firewall/packet_filter.htm, September, 2000.

[9] Curtin, M. and M. J. Ranum. Internet Firewalls: Frequently Asked Questions, rev 10.0. http://www.interhack.net/pubs/fwfaq, December, 2000.

[10] GNU/Linux, v2.4.x. http://www.kernel.org

[11] OpenBSD, v2.8. http://www.openbsd.org

[12] Graichen, T. "Performance Comparison and Tuning of Free Operating Systems". http://innominate.org/~tgr/slides/performance, 2000.

[13] Bugtraq vulnerability mailing archive, SecurityFocus.com. http://www.securityfocus.com/bugtraq/archive

[14] Computer Emergency Response Team (CERT). http://www.cert.org

[15] System Administration, Networking and Security Institute (SANS) Global Incident Analysis Center (GIAC). http://www.sans.org

[16] Squid Web Proxy Cache. http://www.squid−cache.org

[17] Firewall Tool Kit, Trusted Information Systems, Network Associates, Inc. http://www.fwtk.org/main.html

[18] Gauntlet Firewall, PGP Security, Network Associates, Inc. http://www.pgp.com/products/gauntlet

[19] Tripwire commercial software. http://www.tripwire.com

[20] GNU General Public License. The GNU Project. http://www.gnu.org/philosophy/license−list.html

[21] GPL Tripwire project, v2.2.1 for Linux. http://sourceforge.net/projects/tripwire

[22] Advanced Intrusion Detection Engine (AIDE) v0.70. http://www.cs.tut.fi/~rammer/aide.html

[23] Syslog−ng, v1.4x. http://lists.balabit.hu/products/syslog−ng

[24] Stunnel Universal SSL Wrapper, v3.14. http://www.stunnel.org

[25] Logcheck, v1.1.1. http://www.psionic.com/abacus/logcheck

[26] Hansen, S.E. and E.T. Atkins. "Centralized System Monitoring with Swatch". 1993 LISA Conference. Usenix Association. http://www.stanford.edu/~atkins/swatch/lisa93.html

[27] Pretty Good Privacy (PGP), PGP Security, Network Associates, Inc. http://www.pgp.com

[28] GNU Privacy Guard, v1.04. http://www.gnupg.org

[29] OpenSSH, v2.52. http://www.openssh.org

[30] OpenSSL, v0.96a. http://www.openssl.org

[31] Kent, S., and R. Atkinson. "Security Architecture for the Internet Protocol". RFC 2401. November, 1998.

[32] Hamzeh, K., et. al. "Point−to−Point Tunneling Protocol". RFC 2637. July, 1999.

[33] Lee, W., C.T. Park, and S.J.Stolfo. "Automated Intrusion Detection Using NFR: Methods and Experiences". In Workshop on Intrusion Detection and Network Monitoring (ID ’99) Proceedings, pages 63−72, April, 1999. Usenix Association. Berkeley, CA.

[34] Snort Lightweight Intrusion Detection for Networks, v1.7x. http://www.snort.org

[35] Forster, J. Snort Ruleset Database. http://www.snort.org/Database/rules.asp

[36] Analysis Console for Intrusion Detection (ACID), v0.9.6x. http://www.cert.org/kb/acid/

[37] Tcpdump/Libpcap. http://www.tcpdump.org

[38] Ethereal, v0.8.17. http://www.ethereal.com

[39] Applied Geographics, Inc. http://www.appgeo.com

[40] ExploreZip.worm. http://vil.nai.com/vil/dispVirus.asp?virus_k=10339

[41] Red Hat Linux 6.x. Http://www.redhat.com

[42] Linux IP Firewalling Chains, v1.3.10. http://netfilter.filewatcher.org/ipchains/

[43] PoPToP, v1.0.1, The PPTP Server for Linux. http://poptop.lineo.com

[44] Linux FreeS/WAN, v1.9. http://www.freeswan.org

[45] IPTables FTP Stateful Inspection Arbitrary Filter Rule Insertion Vulnerability. Bugtraq ID #2602. http://www.securityfocus.com/bugtraq/archive