Proper Care and Feeding of Firewalls

Alec Muffett
Sun Microsystems, UK
*alec.muffett@sun.co.uk**

November 13, 1994

Abstract

With the recent astounding growth and popular appeal of *The Internet*, network security is becoming increasingly important. Many sites are re-evaluating their security policies, and reviewing what defenses can be built, to keep the "teeming hordes" of hackers at bay.

This paper reviews issues relating to one such defense – proper selection, construction, and maintenance of an *Internet Firewall* – a network node connected between a site and its' network peers, providing a configurable gateway through which *all* site-related network traffic *must* pass, in either direction.

Introduction

There are few hard rules about what constitutes a firewall – certainly a firewall is an apparatus, placed onto your network with the intention of permitting "allowed" users to go about their business without obstruction, whilst keeping out hackers, nosey-parkers and other "not-allowed" undesirables.

However, the above attempt at *defining* a firewall encompasses a broad expanse of form, function, obtrusiveness – and technology.

Depending on the size and requirements of your institution and your userbase, your firewall may comprise anything from a few filtering rules hacked into the configuration of your local router, to half a dozen dedicated workstations running on a private network with filtering chokes between you & it, and between it & the Internet.

Throw in a selection of digital token authentication devices ("*Smart Cards*"), switched ethernet hubs, point-to-point encryption devices, pro-active sniffers, tripwires, alarm systems and radio pagers, and the cost of creating this network guard-dog begins to skyrocket.

At the other end of the scale, a perfectly good packet filtering firewall *can* be created for the price of a cast-off PC-clone, a couple of ethercards, and some free bridging software taken from the very same Internet that you're trying to protect yourself from.

Security, as ever, is a matter of what you're willing to pay versus what risks you are willing to accept, taking into account what additional benefits might be gained from your actions.

With this in mind, let us ask the question...

Why Firewall?

- Security
- Policy
- Auditing

The prime excuse for wanting or installing a firewall is usually "to improve security". As the term descends into buzzwordom, installing a *Firewall* – the act of *Firewalling* – will be considered a panacea for all security woes, as if installing a firewall is a trivial undertaking that will protect your network as securely as a pack of Rottweilers would guard your home.

However, the similarity doesn't end there.

As with keeping a Rottweiler, maintaining a firewall is easy, but it is a responsible job which requires discipline if you're going to do it properly.

*also: *alec.muffett@uk.sun.com* and *alec@hicom.org*

A firewall needs continuous care and attention, occasional exercise, and regular check-ups – and if you don't look after it properly, then one day, you are going to get bitten.

The demands that are made on a firewall administrator vary greatly, dependent upon the sort of firewall installed, and the purpose that it serves.

If, for instance, the firewall is nothing more than a filtering router meant to prevent unwanted traffic from sneaking onto a subnet which contains (say) a University's student database records, then the filtering rules can probably be made very simple (read: *draconian*) without causing too much trouble, and maintenance issues are few, if the perceived threat is slight.

A large firewall protecting a campus-sized network is a different matter. Some sites can measure their daily network traffic to/from the Internet in gigabytes (perhaps terabytes?), and the firewall design must be capable of policing *all* this traffic and logging any anomalies that it finds. The logs themselves may run to several hundreds of megabytes daily.

So what does a good firewall provide you with?

Security A firewall gives its administrator the ability to selectively permit or deny access to his network on the basis of the protocol used, host or network addresses involved (*both* destination *and* source addresses), or any other pertinent control such as time-of-day or the method of authentication used to create the connection.

Policy Although this sort of control is sometimes ethically questionable or functionally impossible to achieve *perfectly*, a firewall can be used to enforce restrictions upon outwardly-bound traffic, eg: restricting access (perhaps by particular users?) to certain external hosts or sites, or preventing passage of particular protocols through the firewall, or restricting bandwidth consumption of the same.

Auditing Finally, firewalls are usually in an ideal position to gather usage statistics; all traffic must pass through them, and therefore (if you can cope with logging that amount of data), you can track usage of the network link on an hour-by-hour basis, analysing what protocols cause the greatest hit on "throughput".

The results from boiling-down such statistics are often surprising, and can lead to many cost-saving optimisations and bugfixes – not to mention their other use: heavy artillery in the war to get funding for a network upgrade.

There are other benefits of course, ones which fit into zero or more of the above categories, but in summary it is fair to say that a sensibly chosen and properly installed firewall is a resource which can provide you with many more practical (and sometimes unexpected) facilities, other than just a fuzzy feeling of "security".

So, we can now ask the question:

What Firewall?

- Packet Filtering Gateways
- Circuit Gateways (or TCP Tunnels)
- Application Gateways

As hinted above, there are many types of firewall, each of which provides a distinct aspect of firewall functionality, and a particular style of access control. These basic types can then be combined to provide more complex or refined access control and accountability.

At the "lowest" network level, we have the *packet filtering gateway* – an item of hardware placed in a network acting as either a router or a bridge, restricting traffic flow according to a set of "filtering rules".

Packet filters are expected to have at least a basic understanding of the transport protocols that they will see "coming up the wire" and thus the filter rules can typically control traffic on the basis of:

transport endpoints – or a notion of what's *inside* and what's *outside* the network. In the TCP/IP world, this is usually implemented by masking off portions of the source and destination addresses, and checking whether or not the remaining parts of the addresses refer to hosts inside the secured network.

transport protocol – for instance, TCP, UDP, or raw IP. Other protocols may or may not be directly supported, or it may be assumed that they are to be tunneled through the firewall.[1]

protocol options – a good firewall should have the ability to "drop" traffic on the basis of protocol-dependent options which might compromise security if misused – for instance, the IP "source routing" option which can be utilised in traffic forgery.

Similar issues arise when trying to ensure proper handling of eg: ICMP packets – control messages necessary for the proper operation of IP.

connection endpoints – the most critical facility in a packet filter is the ability to match network traffic against a table of permitted source and destination hosts (or networks), but it is also vitally important to note that the firewall's checking *must* be done against both ends of a connection, and must take into account the service port numbers at each end of the connection, otherwise the firewall may be trivially subverted[Cha92].

Connectionless protocols[2] like UDP cause so many headaches in this area that it is not uncommon for packet filtering firewalls to drop *all* UDP traffic, regardless of destination.

This step appears exceptionally draconian at first glance, since it blocks access to some network services such as WAN-wide NFS, or use of "Archie" via its' *Prospero* interface (via *xarchie* et al.); on the other hand it protects your site from:

- NFS attacks caused by misconfiguration
- NFS attacks caused by bugs
- TFTP sniffing attacks
- DNS attacks on machines *inside* your network
- FSP traffic (FTP-like program, popular with software piracy addicts)
- Routing Protocol (RIP) spoofing attacks
- attacks upon *some* common-but-insecure RPC based services[3]

– given this, the fact that NFS is not necessarily the best way to transfer files across the Internet, and fact that that there is more than one way to access "Archie", dropping all UDP traffic actually appears quite beneficial.

A filtering gateway is usually at the core of any firewall system which aspires to provide its users with transparent access to the Internet, without requiring the use of "custom" applications for accessing non-local network resources. Packet filters are also frequently used as building blocks in the creation of more complex firewalls, in combination with other technologies, such as *circuit gateways*.

Circuit gateways depend on the operation of (usually) a single machine, connected to – but not routing traffic between – both the inside and outside networks.

Typically, this machine will continuously run one or more *proxy clients* – processes which field connection requests from user programs (eg: a specially modified *telnet* program), and transparently make the connection to the appropriate machine *on the other side* and passes data back and forth across the link, so that the user sees no practical difference between what he is doing (a firewall-enabled telnet session), and the real thing (an ordinary telnet session).

The "proxy" process has no understanding of protocols which might be traveling up and down the virtual circuit thus created. The data merely travels in a "tunnel" from source to destination, through the firewall. It is this property which distinguishes circuit gateways from *application relays*, the third major type of firewall.

In an application relay, we again have a situation where a proxy clients of some description run

[1] for example, an AppleTalk, Novell, or X.25 link might be run across the Internet by encapsulating the traffic at one end, encrypting it, and then passing it through a TCP circuit to the remote end, for decryption and injection into the remote network. For obvious reasons, this is known as *tunneling*.

[2] *Connectionless* protocol are ones which exchange messages between client and server hosts without first requiring the client to establish that the server can/will communicate back to the client.

This is opposed to *connection-oriented* protocols (like TCP) which have the notion of a "call setup phase" – and it is a firewall's ability to meddle with the setting up of the connection which permits it to censor communications. Hence connection*less* protocols are tricky to deal with.

[3] nb: only *some*, because many RPC services can utilise TCP as an alternative transport protocol

on the firewall machine, but instead of creating a virtual circuit between source and destination, these clients field the *protocol* that is being utilised, and must maintain some sense of *state* of the connection; a commonplace example is the familiar "SMTP Mail Gateway" – a central machine which receives all incoming and outgoing e-mail and processes it, stripping and removing headers, and so on, before delivering it to its intended destination.

A more complex example would be that of an FTP proxy client – the proxy service must catch the FTP client's *PORT* commands [PR85] which contain the client's IP address and port number, etc, create an equivalent TCP port on it's *own* host, and pass the description of the latter onwards to the FTP server.

When the server "calls back" to the port on the firewall, the data that the firewall receives must be passed back to the client, on the original port that *it* is expecting traffic to arrive on.

This task is evidently much more complex than the provision of a simple point-to-point TCP circuit; unfortunately it is the nature of many TCP services to require this sort of nannying if they are to pass through a firewall, and (looking on the bright side of the situation) the installation of a firewall with protocol-specific proxy services yields a wonderful opportunity to "enhance" the functionality provided by these protocols; for instance, on or near a firewall is usually the most sensible place to install a caching HTTP daemon (a hypertext server for The World Wide Web), to reduce traffic on long-haul WAN links and improve client latency.

Application proxies are also usually the best source for statistics on application and network usage, as cited above.

These latter two kinds of firewall are those most often utilised to provide services going *through* the firewall, either outgoing, or incoming. The simplicity of their implementation is attractive – a few lines of code to make the remote connection and to copy data back and forth, perhaps running out of *inetd* to save hassle, and protected by TCP Wrappers.[Ven92] Such relays are often placed on one or the other sides of a packet filter, and the traffic that they carry is the *only* traffic permitted through the firewall.

Indeed, this is the basic structure for most forms of "advanced" firewall.

To summarise:

- A firewall is typically composed of three or more technologies; packet filtering, circuit relaying, and application gatewaying.

- A firewall connects two networks in such a way that zero or more types of network traffic may pass directly from one network to the other, and so that (by using special tools) zero or more users on one side of the firewall, may utilise services on the other side of the firewall.

- A *good* firewall provides the users of one network with access to the other network in accordance with the firewall's owner's security requirements.

- A *good* firewall provides the required level of security as above, without hindering network performance (ie: is not a *bottleneck*).

- An *advanced* firewall fulfills all of these rôles, *and* provides its owner with enough logging information to be useful for post-mortem incident analysis work.

How Firewall?

Okay, so, now you've decided to improve security but do not feel up to the task of trying to secure each of the machines under your control, individually. You want to install a firewall.

Where should you go from here?

Before jumping into the purchase of some commercial firewall technology (for such beasts *do* exist) – or wading through the installation of some freely redistributable package, it is advisable for you to:

- look at your security policy to see what it is that you want out of your firewall.

- look at your network and the services that you provide to your users, so that you do not incapacitate them.

- look at your existing hardware, to see if it has any unused capabilities which could be utilised.

So, for instance, if there is only a single subnet which requires firewalling, perhaps the judicious installation of a few packet filtering rules into a nearby bridge or router is required, with inbound traffic only being permitted to one, particular, host, which is then further protected by application wrappers[4].

On the other hand, if the entire campus[5] network needs blanket security, your best bet is to move the packet-filter up a few levels, onto a dedicated piece of hardware, with spares and backup facilities as required.

Firewall selection should be dictated by functional requirements and the amount of control desired. For those who are chiefly interested in examining the issues relating to design, or who would like to examine the schematics of some real-world firewalls, several excellent technical papers and books exist [Ran93] [CB94] – but, to the intrepid administrator who prefers to choose a firewall "off the shelf", most of the issues are easily resolved by application of decisiveness, and a little common sense.

Probably the most important points are:

Functionality – it cannot be emphasised enough that you must look long and hard at what functionality your users want[6], and decide what functionality *you* wish them to have, and whether to be *inclusive* (anything not explicitly forbidden is permitted) or *exclusive* (anything not explicitly permitted is forbidden).

– then choose your firewall design to fit these requirements.

Simplicity – the most flexible and robust firewalls are those which are simple. Preference should be given to "all or nothing" type restrictions – and where security is concerned, "nothing" is often far better than the alternative.

Designs which strive for simplicity are easier to recover or rebuild, should disaster strike; "simple" designs can also be built upon easily (although this makes them less simple) in order to cope with extraordinary circumstances, and (most importantly) "simple" designs are hard to foul up; admittedly, there are always subtleties which must be taken care of, but this is a fact of life, and most of the pitfalls can be avoided by the methodical administrator who reads around the subject.

The essential point is that with fewer "exceptional" rules, there are fewer opportunities for mistakes which could compromise security. Hence the reason for common design decisions such as:

- firewall accepts no logins from "outside"
- firewall has no ordinary "user" accounts
- firewall uses digital token authentication (or similar)
- firewall uses static routing tables
- firewall ignores DNS or utilises reverse-lookup security checks
- firewall does not use NIS/NIS+, and neither trusts nor depends on any other machine
- firewall drops all source-routed IP traffic
- no UDP traffic is permitted through the firewall
- all outgoing TCP connections are permitted
- no incoming TCP traffic is permitted, except:
  - SMTP to the mailhost
  - WWW to the HTTP daemon
  - Gopher to the Gopher server
  - FTP to the Anonymous FTP server (a machine with no user accounts)
  - Telnet to a host with digital token authentication (or similar)

...are common. The aim should be to create restrictions which are simple to describe and simple to implement.

Isolation – in high-security firewalls, another aim should be that of service isolation; if a machine providing a particular service is subverted (eg:

[4] No-one has established the diminutive of "firewall" in the network sense, but "fire-ridge" is a popular choice to describe this setup. Some have frivolously suggested "fire-speedbump" for even smaller firewall applications – not my suggestion, I might add. `8-)`

[5] where *campus* refers to any institution occupying several buildings.

[6] Installing a firewall is when you *really* get to learn what your users are up to.

a machine which provides telnet gatewaying), the fact that it has been subverted should not[7] impact the security of other services.

Security – finally, the machines you use to implement your firewall should be as secure as you can make them.

This includes not only the obvious requirements, such as restricting "login" access to the "console" (or equivalent) and removing un-necessary functionality (superfluous network services, compilers, user accounts and user applications such as X-windows), but security also includes matters of administrator discipline, such as monitoring and acting upon the reports from logfiles, archiving old reports, examining suspicious activity, and maintaining write-protected "virgin" copies of the software installed, in case of a breakin.

In fact, the matter of a firewall's integral security really deserves a section to itself, so...

"Care and Feeding"...

...or "Keeping your firewall safe from harm".

Having delegated a large portion of the responsibility for network security to a group of one or more machines, it seems only sensible that those machines should be kept safe from harm.

Frequently the hardware involved will be too "stupid" to be worth worrying about (eg: dedicated single-user PCs locked away from harm), but if a *multi-user* machine is involved, considerably more care must be taken.

Quite often, such a machine will be a "bastion host" – pivotal to your firewall's integrity and directly connected to the Internet, it must be able to withstand everything that the Internet can throw at it.

How can this be achieved?

- Do not build your bastion host once, and then forget about it.

  Maintain communications with the appropriate vendors. Watch for reports of security patches. When patches come out, apply them after confirming with your vendor that they are stable. Be aware that someone might, one day, produce a "trojan horse" patch and try to pass it off as the real thing.

- Be alert, otherwise you might just install a buggy program (eg: an old version of *sendmail*) on your bastion host, and your security would be shot. If you're concerned about something, contact your regional security advisory body for advice.

- Strip your bastion host down to the barest minimum of required functionality. As stated above, all un-necessary software should be removed in order to reduce complication.

  Why risk running possibly dangerous software on the most critical host on your network?

  Why have the software installed on that machine at all?

- Restrict access to the firewall *itself* – only critical systems accounts should be permitted on this machine, and there should be no need for anyone to be able to log onto the machine from anywhere other than "console".

  Change the shells that are associated with passive systems accounts to something harmless like */bin/false* – or better still, install a custom "shell" of some sort that triggers an alarm when invoked, and use *that*.

- Do not restrict yourself to using only "reusable" passwords for authentication; the price of digital token authentication devices (or their software implementations) is dropping slowly and there are viable alternatives available openly on the Internet; eg: Bellcore's S/Key package, which manages authentication via a printed list of one-time passwords.[8]

  The importance of selecting and using a one-time password technology is growing rapidly. There are now frequent newspaper reports of passwords being "sniffed" by hackers using tools such as *tcpdump*, and in the Unix[9] world

[7] for small values of "not".

[8] S/Key is a supported authentication mechanism with the TIS Firewall Toolkit.

[9] Unix is probably still someone's trademark, but I don't know whose.

there is the danger of your password ciphertext being broken by a password cracker.

Methods of authentication which cannot be replayed are a much safer option, nowadays.

- Monitor attempts to use the services you disable, via the TCP Wrappers[Ven92] package, and wire the wrappers to the same sort of alert mechanism as described above.

- Configure your system so that any activities related to security leave some sort of audit trail behind – preferably stored on immutable media, but if this is not possible, then at least try to check that your logfiles only ever grow bigger, not smaller.

- Install auditing software such as "Tripwire' [KS93] or its smaller cousin "L5" which takes a snapshot of your filestore, and then can be run regularly to spot unexpected changes to your filestore.

  Avoid totally automating this process (if possible) lest your system be hacked and your auditing script gets modified so that it never tells you that anything is wrong.

  Another important point about using Tripwire properly is that you should store your snapshot database (or a checksum of your snapshot) on a read-only media; floppy disks are useful for this.

- Log your most critical events to hardcopy if at all possible.

- Finally... CHECK YOUR LOGS FREQUENTLY.

  Your logs are critical. They will mostly contain nothing but (boring) records of common transactions; but – just occasionally – there is the merest *hint* that something is wrong. A warning message that you cannot explain, for whose existence you can find not justification. Something that you've not seen before.

  It is *that* which you must be alert for.

  In fact, there are tools such as "swatch" to help you do this, if it all becomes too much. The sheer repetitiveness of most logging information makes it ideal for automated analysis, as anyone who has ever faced the tasks of analysing system-accounting records or laser-printer usage accounting will be familiar with.

Securing a bastion host not a hard task; it is just a matter of being thorough.

Great Expectations

The past few sections have dealt exclusively with the positive aspects of firewalling; there are negative aspects to a firewall too (repeat after me: "firewalls are not a panacea") and as such, it is probably worth comparing and contrasting firewall usage against other methods of securing hosts on your network.

Disconnection – removing yourself from the Internet is perhaps the ultimate in firewalling, and it certainly improves your resistance to attack from "outside". It is a cheap and simple solution, and is occasionally regarded by people in the higher echelons of management as the preferable (or even *desirable*) security option.

Contrawise, disconnection may also be considered the antithesis of "true" firewalling – the art of maintaining communications with the Internet whilst maintaining security.

It cannot be denied that, to some bodies, Internet connectivity is *not* of truly mission-critical importance, yet it must also be noted that turning your institution into an Information-SuperHermitage can rapidly lead to loss of interaction with the academic and commercial worlds, and eventually to ostracism from the community.

As ever, it is a matter of deciding what you *really* want.

Passwords Password technologies fall into two camps, the *reusable* and the *non-reusable*.

Reusable passwords are the ones most familiar to us, but the reliance that we still place upon them *today* is astonishing, given their epically insecure nature. Stories of their insecurity abound, from childrens stories ("Open Sesame!") to more modern versions of the tale [Sto89].

Password security is doubly irrelevant in this modern age of non-interactive, promiscuous network services – FINGER, SMTP, POP, FTP, NFS, and so on – which have little or no concept of *authentication*, or which trust a users' transactions on the basis that *another* machine has apparently authenticated him.

On an open network as large as the Internet, where anyone could be "hiding behind a rock", listening in on your transactions, to solely rely upon authentication by reusable passwords would be madness.

The solution?

Experiment with non-reusable passwords: "Challenge-Response" systems and "One Time Passwords". Seek out methods of authentication which cannot be recorded and replayed to effect entry into any of your systems. This will not keep some of the more "trusting" network services safe from harm, but it does reduce the threat slightly.

TCP Wrappers The TCP Wrappers package can make a wonderful addition to your security – they bring access control list (ACL) style functionality and extra auditing features to many network services, and as such can implement "firewall-ish" functionality on a per-host basis.

Is there a catch to this, from a broader perspective?

Yes: that you have to implement "firewall-ish" functionality on a *per-host* basis.

This is not to deride the TCP Wrapper software in any way; it really is excellent, and has its own place in security implementations, however it does not necessarily provide a complete security solution, because:

- It *does* require rolling out on every host on your network to provide anything approaching the same level of security that a single firewall can.

  This is a problem of administration, standardisation and scalability, all of which are hard to control in a research environment.[10]

- TCP Wrappers do not necessarily bring you any protection against attacks which 'mess with your machine's mind, like ICMP "nukeing", and routing attacks. The wrappers are designed to protect the higher protocol levels utilised by user applications, rather than kernel-level network protocols.

- By nature, solutions based entirely around TCP Wrappers are *inclusive* – you have no control over those services which you do not "wrap".

  Therefore if you fail to install wrappers on *every* host, or if you fail to wrap a particularly insecure service, you are in just as much danger as before (although perhaps you will be less *exposed* then you would otherwise).

- In a related issue, TCP Wrappers bring you no protection against collusion; your users are able to create (eg:) daemons for WWW or IRC or FSP and publish their addresses to others as an Internet Resource. Permitting your users this degree of freedom may or may not be desirable, depending upon how responsible they are.

  If this freedom is permitted but abused, liability issues may arise if (for instance) a user creates an FSP archive of pornographic material on some otherwise innocuous workstation, is discovered, and brings down the wrath of the tabloid press upon you.

Encryption – Encryption can be another useful addition to your security arsenal; the ability to obscure your traffic as it travels along the wire is useful in some circumstances (when handling financial transactions or other sensitive data) and may be desirable, but is often of limited use in protecting your network from *assault*.

Encryption technology is neither ubiquitous nor (usually) free, and for the moment appears to be geared towards securing communications between two consenting network applications, or to leverage the use of authentication technologies.

[10] ...and yet, researchers are always the ones to scream most loudly, should a hacker happen to trash all their data.

> In open and diverse environments, usage is hampered by licensing problems, and the general lack of availability for any one technology across all platforms. Therefore, the question of whether encryption can be fruitfully used in an environment should really be left to the person who would have to administrate that environment.

We should now ask: "what are the restrictions on firewalls?" – the response is complex, and of course depends greatly upon what sort of firewall you install. Probably the fairest response would be:

> "With the application of a little imagination, there are few restrictions upon what a firewall can be made to do, or what level of security a firewall can be expected to provide.
>
> A firewall provides security on a "put all your eggs in one basket, after building a really *strong* basket" approach to security.
>
> The problem that will most frequently be encountered is that a firewall usually acts as an (almost) passive carrier of network traffic, forbidding or permitting traffic movement on the basis of its apparent source or intended destination, or whether (prima facia) the traffic is utilising some suspicious or unusual protocol options which set it apart from run-of-the-mill data.
>
> Firewalls, by and large, are not equipped to identify security exploitations encapsulated *within* otherwise viable protocol streams."

Most firewalls could not protect against (for instance) an E-Mail message destined for delivery to a valid user, a message which (once past the firewall's gateway mechanism) exploits a bug in the mail delivery agent to compile a small payload program, which does something obnoxious like removes the user's filestore, or which tunnels *back out* through the firewall, providing entry to some band of vandals.

This is why installation of a firewall should not be considered a *complete* solution to all of your security woes. Firewalls can *screen* traffic, and do so very effectively, but they do not give total protection from the contents of the data therein.

Remember too, that the whole point of firewalling is to divide the network into two halves, the **THEM** side of the network, and the **US** side.

A complete security solution would have to address the fact that often, we can't even trust *ourselves* – in any large environment, who can say that they fully comprehend the motives of all of their users. Who can say that all of the people on the the "US" side are completely trustworthy?

A complete solution requires you to undertake security measures at all levels of network usage, from application access (access control and encryption) through the network layer (firewalling, unspoofable routing and snoop-proof networking) to the physical layer (restricting unauthorised connections to your network cable).

If you take network security seriously, you'll have to admit, it's not just a job for a firewall.

References

[CB94] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security*. Addison Wesley, 1994.

[Cha92] D. Brent Chapman. Network (In)Security through IP packet filtering. In *Proceedings of the third USENIX Unix Security Symposium*, 1992.

[KS93] Gene H. Kim and Eugene H. Spafford. The design and implementation of Tripwire: A filesystem integrity checker. Technical Report CSD-TR-93-071, Purdue University, 1993.

[PR85] John Postel and Joyce Reynolds. File transfer protocol. RFC-959, 1985.

[Ran93] Marcus J. Ranum. Thinking about firewalls. In *Proceedings of the Second International Conference on Systems and Network Security and Management (SANS-II)*, 1993.

[Sto89] Clifford Stoll. *The Cuckoo's Egg*. Doubleday, 1989.

[Ven92] Wietsa Venema. TCP WRAPPER: network monitoring, access control and booby traps. In *Proceedings of the third USENIX Unix Security Symposium*, 1992.

Many of the papers cited above are available directly from the Internet. Amongst the most important resources/sites are:

CERT FTP Archive *ftp://cert.org/*

This is the software archive maintained by the American Computer Emergency Response Team, containing many useful security tools, and back-issues of their security advisories.

COAST FTP Archive *ftp://coast.cs.purdue.edu/pub/*

This is the COAST Project archive – probably the most complete archive of computer-security related software ever assembled. All of the packages documented above will be found here, either as master copies, or mirrored from their chief main sites.

Firewalls maillist archive *ftp://ftp.greatcircle.com/pub/firewalls/*

Contains much reference material, and past issues of the digested "Firewalls" maillist. To subscribe to the maillist, send the message:

`subscribe firewalls` *you@your.email.addr*

to *majordomo@greatcircle.com.*

TIS Firewall Toolkit *ftp://ftp.tis.com/pub/firewalls/*

The home of a popular, freely-redistributable set of proxy clients and associated firewalling tools, as well as many technical papers upon the subject.

TAMU Security Toolkit *ftp://net.tamu.edu/pub/security/*

The Texas A&M University security toolkit, comprising a set of security auditing scripts configurable for most popular operating systems, a sniffing/reporting program, and *Drawbridge* – a PC based packet filtering software bridge.

Glossary

archie – an database running on several freely-accessible machines, which stores lists of Anonymous-FTP'able software and other data available on hundreds of archive sites worldwide.

Can be queried through an interactive text-based interface, via WWW software, or by using specialised UDP client software.

DNS – Domain Name System, a distributed, hierarchical directory service, used to map between Internet addresses and hostnames. Utilises both UDP and TCP.

finger – a common service available on most Internet machines, used to provide information about the users of a particular hosts to anyone who wants to know.

FSP – a low-bandwidth UDP-based file transfer protocol which requires no special priviledges to run, with similar functionality to FTP.

FTP – File Transfer Protocol. A standard client/server application used to transfer files between TCP-connected hosts.

Notable for its notion of *Anonymous*-FTP, the ability to configure the FTP server to "publish" data stored in a particular area, so that anyone may retrieve it without requiring an account on the server's host.

Gopher – a TCP based protocol (with associated client software), which allows a user running a Gopher server to "publish" information to the rest of the Internet.

HTTP – HyperText Transport Protocol. The TCP-based protocol used to transfer documents in the World Wide Web hypertext system.

Similar in concept to Gopher, but used for transferring hypertext documents to clients such as "Mosaic", which format the data for presentation to the user.

ICMP – Internet Control Message Protocol. A datagram based network protocol running alongside IP, used to shepherd network connections, report network outages and disconnections, and prevent traffic flooding across slow network links.

In a firewalled environment, ICMP traffic should be dealt with carefully. It is quite simple to forge ICMP messages which will forcibly reset all TCP connections to a particular host, or re-route traffic to flow through a third-party host for "snooping" purposes.

IP – Internet Protocol. A low-level host to host, datagram-based connectionless protocol, used as the standard for communications between Internet-connected sites.

IRC – Internet Relay Chat. A TCP-based, distributed, erratic world-wide "chat" system where thousands of people congregate to discuss, argue, and whinge about trivia.

NIS – Network Information Service. Distributed network database software, typically used to provide Unix password, group, and network information, to disparate network hosts, in a campus-like environment.

NFS – Network File System. A RPC-based protocol which permits sharing of files from a server machine to hosts on a IP-based network.

POP – Post Office Protocol. A TCP-based service which permits a user to download waiting E-Mail, for processing on a PC or laptop computer.

RIP – Routing Information Protocol. A UDP-based protocol used by Internet hosts and routers to exchange and maintain routing tables – information which describes the best route for traffic to take, through the network, in order to reach its intended destination.

RPC – Remote Procedure Call. RPC is a library of software routines which permit programmers to create applications which work in a distributed manner.

Such RPC applications use either TCP or UDP to pass data to a service running on a remote host; the service then processes the data and returns the result to the user. NFS, for instance, is implemented as a library of RPC calls.

SMTP – Simple Mail Transfer Protocol. A TCP based protocol which is used to exchange E-Mail between internet-connected sites.

TCP – Transmission Control Protocol. A connection-oriented, virtual-circuit communications protocol, typically layered on top of IP, used to provide reliable and unduplicated communications between Internet-connected hosts.

telnet – a TCP-based "Virtual Terminal" protocol, which provides interactive access to a remote computer across the Internet, as if connected by a serial line.

TFTP – Trivial File Transfer Protocol. A simplistic, UDP-based file transfer protocol, commonly used as part of the bootstrap process in booting diskless workstations. Notable because (if misconfigured) it will transfer any world-readable file on a Unix system, to anyone who asks for it.

UDP – User Datagram Protocol. A unreliable datagram service layered on top of IP, which provides a data checksum and an application port number on top of the basic IP services.

Unlike TCP traffic, each UDP datagram stands alone, and is delivered to its destination without any prior negotiation. It is this feature of UDP which makes it particularly hard to guard against forgery of UDP traffic.

WAN – Wide Area Network. Any large area network spanning political boundaries.

WWW – World-Wide Web. An Internet hypertext system in which publically accessible hypertext daemons can be queried using the HTTP protocol, to return all manner of information or services, back to the user.

Author

Before joining Sun, Alec graduated from UCL and spent three and a bit years at the University College of Wales, Aberystwyth, working as the Computer Unit's first Unix Systems Programmer.

He now works for *Sun* as the European representative of their *Network Security Group*, responsible for policing and auditing Sun's internal network, evaluating security products and architectures, and incident handling.