The ISP Column
A monthly column on all things Internet

Revisiting Trust

April 2003

Geoff Huston

In December 2001 I wrote an article that explored the notions of trust within the design of Internet services. It seems that trust, and the associated topic of security is becoming an increasingly important topic, so I'd like to revisit that article and explore some of the issues in further detail in this column.

A major current focus for the Internet is one of security. The fascination over the accelerating growth curves of the industry, and the compression of technology lifecycle time scales into "Internet years", coupled with the constant enthusiastic search for the next "killer app" appears to be over. This is probably cause for a collective sigh of relief, as I'm not sure how long the industry, let alone the individuals working in the industry, could sustain such intense effort and such rapid pressures for change.

In its place there appears to be a period of consolidation of the services used upon the Internet, and this is probably a timely and necessary activity. A such a phase of consolidation it is appropriate to look again at the services and facilities that populate the Internet and ask the question as to how resilient such services are. Are they robust? Can they be trusted? Are the service edifices we've constructed on this Internet base anchored in a solid foundation? The Internet is assuming an ever-broader role of underpinning all kinds of commercial, governmental and personal activities, and its important to all that it operates as reliable and trustable service.

Lets start with the foundations of the Internet. How robust is the Internet? How resilient is the Internet to attempts to disrupt it?

Interestingly enough the early work in packet switched networks was focused on exploring the notion that it could be possible to construct a system that was more reliable than its components.

> The Internet is undoubtedly the unintended outcome of the initial research objectives articulated by the Advanced Research Projects Agency (ARPA) of the United States Department of Defense (DoD) in the late 1960s, but the antecedents of this effort go back a few years earlier in the U.S. research community. The so-called think-tank of the cold war, the RAND Corporation, was an early vehicle for the concept of computer networking. There, Paul Baran, whom many consider to be the father of computer networking, presented his ideas on the subject in a seminal work published by RAND in 1964, "On Distributed Communications"

"...one day we will require more capacity for data transmission than needed for analog voice transmission. ... it would appear prudent to broaden our planning consideration to include new concepts for future data network directions. Otherwise, we may stumble into being boxed in with the uncomfortable restraints of communications links and switches originally designed for high quality analog transmission. New digital computer techniques using redundancy make cheap unreliable links potentially usable. A new switched network compatible with these links appears appropriate to meet the upcoming demand for digital service. This network is best designed for data transmission and for survivability at the outset."

"On Distributed Communications: I - Introduction to Distributed Communications Networks," RM-3420-PR, August 1964

The entire set of Baran's classic RAND papers on packet-switching from the early 1960s are now available online at www.rand.org/publications/RM/baran.list.html.

However it is perhaps only in theory that there is no difference between theory and practice. The practical experience of the Internet reveals a disturbingly rich history of disruptive attack. Perhaps the most noteable early recorded attack was the "Internet Worm" of the late 1980's which exploited a bug in a number of commonly deployed utility programs on Unix hosts.

This worm had a number of attack modes, including exploiting weaknesses in the sendmail daemon as well as the "finger" daemon. The finger daemon did not protect itself against buffer overflow. Since then documentation of the "gets" library routine includes dire warnings that this is a dangerous function and should not be used

"This is a _dangerous_ function, as it has no way of checking the amount of space available in BUF. One of the attacks used by the Internet Worm of 1988 used this to overrun a buffer allocated on the stack of the finger daemon and overwrite the return address, causing the daemon to execute code downloaded into it over the connection."

[gets manual page, FreeBSD v4.7 documentation]

A copy of the paper on the worm can be found at http://world.std.com/~franl/worm.html

Since then we've seen a steady stream of attacks on the Domain Name Service, attacks on hosts using TCP SYN flooding, viruses, distributed denial of service attacks, attacks on the routing system, Code Red and its variants, and the recent Slammer attack on SQL servers, to point to but a few. Why have we been unable to eliminate such attacks on the network and its attached host systems?

One line of reasoning is borrowed from work in formal systems some seventy years ago. At that time the discovery was made that any formal system that was sufficiently expressive to be 'useful' was sufficiently expressive to contain paradoxes. In a very informal sense any sufficiently powerful

consistent and decideable formal system contains the seeds of its own demise by admitting undecideable propositions into the system!

> The Austrian mathematician Kurt Gödel (1906 - 1978) is best known for his Incompleteness Theorems. In 1931 he published these results in "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme" (On Formally Undecideable Propositions of Principia Mathematica and Related Systems) He proved fundamental results about axiomatic systems, showing in any axiomatic mathematical system there are propositions that cannot be logically proved or disproved within the axioms of the system. In particular the consistency of the axioms cannot be proved.
>
> This placed a full stop on a hundred years of attempts to put the whole of mathematics on an axiomatic basis. One major attempt had been by Bertrand Russell and Alfred North Whitehead with Principia Mathematica (1910-13), as well as work on the same topic by David Hilbert. The guiding principle in this philosophy of logistics was to demonstrate the manner by which all of mathematics could be derived in a deterministic and decideable manner from a set of basic axioms and rules of logic. Further developments focussed on the best way to do this, including efforts to guarantee that one would not find any contradictions, nor any undecideable propositions.
>
> Gödel's work demonstrated that any consistent formal system that is sufficiently powerful to include formal propositions that describe properties of other formal propositions also admit the expression of formal propositions that are themselves undecideable.
>
> http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Godel.html

Perhaps, in the combination of complex computing systems coupled with a rich communications capability, we have developed a system where the components are sufficiently complex that they are vulnerable to attack by virtue of their complexity alone. While such an observation is tempting, it is stretching the analogy well beyond breaking point, and, personally, I do not believe that this view is either useful or accurate. The vulnerability of the Internet is not in the complexity of its components but in the trust model of the Internet.

So lets look at the trust model of the Internet and see what we might do to improve it.

All networks have an inherent trust model. When two parties cannot directly interact with each other to confirm their identity, then they are forced to place some element of trust in the intervening network. In some cases the trust model relies on the ability of the network provider to operate the network correctly. For example when I enter a string of digits into the numeric pad of my telephone I'm trusting the network to create a connection to a particular handset. In other cases the trust model relies on each user acting appropriately, and the network provider is in no position to enforce such user behaviour.

At a basic level the Internet is a simple datagram delivery network. The network itself does not define any service other than packet delivery. Surrounding the network is a collection of end-systems. It is these systems that define the service model of the Internet, and define how the

network resources are to be shared across competing demands. Some of these systems are operated by the same entity that manages the network, but the overwhelming majority of such systems are operated by other users of the network. In the Internet's trust model every user trusts, to some extent, the benign intent of every other Internet user. In a closed homogenous small community this trust may be well placed. In the broader context of a public utility system with hundreds of millions of users, this is stretching the friendship model a little bit beyond its natural limits.

To illustrate this distributed trust model, lets look at the action of download a web page using a browser.

In this example I've collected the IP packets from a local host as it retrieves the web page http://www.isoc.org. The output here is collected from the very useful tcpdump program (http://www.tcpdump.org/)

Once the URL is entered into the browser, the browser's first task is to translate the domain part of the URL into an IP address. The browser passes the domain name to the local DNS service routines and awaits a response. The local DNS service routine has been configured with the IP address of a remote resolver. The IP address of this DNS resolver has been provided automatically as part of the initial connection setup using the DHCP protocol, or it's been manually configured into the local system. The local host asks its DNS resolver to translate the name to an IP address.

The local host is a Windows XP operating system, with IPv6 enabled

The Windows XP TCP driver appears to want to establish a IPv6 connection with the ISOC web server, if it can be achieved, and if not, then drop back to establishing an IPv4 connection

So, using an IPv4 UDP transport, the client queries the local server for a V6 address for the domain name, "www.isoc.org"

This is a tcpdump trace of the IP packet. The packet trace shows that the packet was sent from the source address of 1029 from the system "gih.example.com" and the packet was directed to a local DNS name resolver, "named.example.com", using port 53 on that system. The payload of the UDP packet is a DNS query, with a request for an AAAA DNS Resource Record for the name "www.isoc.org"

```
  gih.example.com.1029   >   named.example.com.53:      16+   AAAA?
www.isoc.org.
    (30) (ttl 128, id 311, len 58)
```

The response is that while the domain exists, there is no V6 address associated with the name

```
  named.example.com.53  >  gih.example.com.1029:    16  q:  AAAA?
www.isoc.org. 0/1/0 ns: isoc.org.
    SOA info.isoc.org. admin.www.isoc.org. 2002121713 10800 3600
604800 86400
    (77) (ttl 60, id 54954, len 105)
```

As the name specified did not include a trailing '.' to indicate a fully qualified domain name, the name resolver system uses the local default domain name suffix to see if there is a V6 address associated with this local name zone (www.isoc.org.example.com)

```
  gih.example.com.1029  >  named.example.com.53:     17+  AAAA?
www.isoc.org.example.com.
    (42) (ttl 128, id 312, len 70)
```

This time the response is different, indicating that there is no such domain name

```
  named.example.com.53 > gih.example.com.1029:   17 NXDomain q:
AAAA?
  www.isoc.org.example.com.
    0/1/0     ns:     example.com.     SOA     dns0.example.com.
hostmaster.example.com.
    2003042204 10800 1800 7200000 3600
    (94) (ttl 60, id 55078, len 122)
```

Now that the V6 queries have failed, its time to try V4, and ask for an address record (or A record) from the DNS:

```
  gih.example.com.1029   >   named.example.com.53:      18+   A?
www.isoc.org.
    (30) (ttl 128, id 313, len 58)
```

This time the DNS returns the address record of 206.131.249.182

```
  named.example.com.53  >  gih.example.com.1029:     18  q:  A?
www.isoc.org. 1/3/2
    www.isoc.org.    A    206.131.249.182    ns:    isoc.org.    NS
info.isoc.org., isoc.org. NS ns.uu.net.,
    isoc.org.  NS  ns.isi.edu.  ar:  ns.isi.edu.  A  128.9.128.127,
info.isoc.org. A 206.131.249.182
    (144) (ttl 60, id 55099, len 172)
```

So the answer is that the domain name www.isoc.org maps to the IP address 206.131.249.182. But can we trust this answer?

The DNS is a highly distributed database, where various components of the database are operated by a diverse collection of operators. It would be comforting to believe that the DNS provides accurate answers to queries all of the time. This is not the case. It would possibly be an acceptable compromise to believe that all incorrect answers are the result of temporary faults or inadvertent operator errors. Unfortunately even this is not the case all of the time. The DNS is a target of various forms of attack, and in some cases such attacks are successful. In such cases the DNS provides incorrect answers, directing the user to a site that may then attempt to compromise any ensuing transaction.

It could be that the name resolver has been compromised, and is delivering incorrect answers. It could be that the name resolver is using a forwarder, and this forwarder has been compromised. It could be that the primary zone for isoc.org has been updated and the secondary servers for this zone have not yet refreshed their local copy. It could be that some part of the routing system has been compromised and DNS traffic addressed towards the authoritative root DNS servers is being redirected towards a different server that is providing incorrect answers, while masquerading as the authoritative root server. It may be that the DNS servers are using some form of load distribution technique to spread the query load over a number of servers, and the distribution mechanism has been compromised. And no doubt there are many more uncomfortable vulnerabilities in the operation of the DNS.

The inevitable conclusion is that DNS is not a secure protocol. There is no means of ensuring that the data one gets back is authentic. As noted in a recent threat analysis of the DNS there are a number of attack modes for the DNS which are undetectable by the victim.

This analysis was undertaken by Derek Atkins and Rob Austein in 2001 within the IETF DNS Extensions Working Group.

The threat modes listed in this study included packet interception, where the simple unencrypted two packet exchange protocol of the DNS makes the system vulnerable to various forms of man-in-the-middle attacks, including eavesdropping with false responses injected back into the network in advance of the actual responses. It is also possible to apply various name games to DNS responses. The DNS relies on extensive use of caching of answers to improve its performance and reduce load on the DNS servers. One form of attack is to feed false data into the cache, which in turn trigger the local system to then query the fake DNS servers. The threat is based on passing back to the query point a DNS response that includes one or more DNS resource records in the Additional section of the DNS response.

As Atkins and Austein pointed out: "The common thread in all of these attacks is that response messages allow the attacker to introduce arbitrary DNS names of the attacker's choosing and provide further information that the attacker claims is associated with those names; unless the victim has better knowledge of the data associated with those names, the victim is going to have a hard time defending against this class of attacks.

This class of attack is particularly insidious given that it's quite easy for an attacker to provoke a victim into querying for a particular name of the attacker's choosing, for example, by embedding a link to a 1x1-pixel "web bug" in a piece of Text/HTML mail to the victim."

A current work-in-progress that describes DNS threats in further detail is by Faltstrom and Mealling and can be found at draft-ietf-enum-rfc2916bis-05.txt.

Once the DNS returns an IP address, the next step is to open an HTTP session. The first step is to send a TCP SYN packet to the IP address to start the connection.

Send a TCP SYN packet to port 80 of the server

```
gih.example.com.1044 > www.isoc.org.80: S 72886617:72886617
  (0) win 64240  (DF) (ttl 128, id 314, len 48)
```

The server should accept the connect request by responding with an ACK of our SYN packet, plus a SYN packet of its own, which it does.

```
www.isoc.org.80 > gih.example.com.1044: S 2174176570:2174176570
  (0) ack 72886618 win 31740  (DF) (ttl 47, id 24462, len 48)
```

We complete the handshake by sending an ACK of the server's syn packet. The TCP session is now connected

```
  gih.example.com.1044 > www.isoc.org.80: . 72886618:72886618(0)
ack 2174176571 win 64860 (DF) (ttl 128, id 316, len 40)
```

How does the user know that the packet is being delivered to the correct server?

Here the user is trusting the integrity of the Internet's routing system. The routing system is vulnerable to incorrect routing information being injected into the routing system, and as a consequence packets may be misdirected. While the majority of such incidents are attributable to operator error, there remains the potential that the routing system is vulnerable to deliberate attack.

Routing can be described as a distributed computation that generally uses some form of a relaying messaging protocol, where route information is received, processed and forwarded. Current routing protocols typically contain no specific mechanisms to prevent the unauthorized modification of the information by a forwarding agent, allowing routing information to be modified, deleted or false information to be inserted without the knowledge of the originator of the routing information or any of the recipients. There is also the potential for interception, hijacking, and denial of service to disrupt the operation of the routing system.

> A current work-in-progress, "Generic Threats to Routing Protocols" by Murphy, Beard and Yang enumerates a relatively expansive set of set of threats to routing protocols. (draft-ietf-rpsec-routing-threats-00.txt)

Of course even if the routing system is operating correctly, the end user is also trusting the integrity of the various forwarding devices along the path to ensure that the packet is being delivered to the correct server in accordance with the local forwarding decision as determined by the routing system. Again this trust may be misplaced.

There is also the assumption that the packets are not being inspected while in flight, with the potential for fake responses to be generated before the remote system can respond. Again, it is not clear that such trust in the confidentiality of data transactions is well placed all of the time.

It could be that the packet is being passed through a so-called "transparent proxy cache". In such a case the packet exchange will look precisely the same, and the received packets will use the source

address of the target server, but the forwarding system is directing the clients packets into the cache server and the cache server is masquerading as the actual server. Has the cache server been compromised? Is the cached copy of the web page an accurate mirror of the current contents of the original or has it been altered in some way?

This initial packet exchange provides no reassurance that the connection being set up is the intended connection with the server that is hosting the desired web page.

When the TCP session is opened, the user's HTTP session then requests the web object from the server.

```
gih.example.com.1044 > www.isoc.org.80: P 72886618:72886974(356)
  ack 2174176571 win 64860 (DF) (ttl 128, id 317, len 396)
```

Hang on - what was in that payload?

```
  GET./.HTTP/1.1
  Accept:.image/gif,.image/x-
xbitmap,.image/jpeg,.image/pjpeg,.application/vnd.ms-
excel,.application/vnd.ms-
powerpoint,.application/msword,.application/x-shockwave-
flash,.*/*
  Accept-Language:.en-us
  Accept-Encoding:.gzip,.deflate
  User-Agent:.Mozilla/4.0.(compatible;.MSIE.6.0;.Windows.NT.5.1)
  Host:.www.isoc.org
  Connection: Keep-Alive
```

Chatty isn't it? Identifying the user agent is perhaps being a little too chatty. If the version of the user agent has a know exploit then telling every server that you are vulnerable is perhaps extending trust too far. It also appears to be letting the server know what applications can be fired up within a response. Again not exactly reassuring information to be passing around to strangers if there is a means of using web response that can trigger a vulnerability within one of these applications.

Of course it could include even more information. When a link is being followed then the request also includes a Referer field, referencing the URL of the page containing the link. The Referer header allows reading patterns to be studied and reverse links drawn. Although it can be very useful, its power can be abused if user details are not separated from the information contained in the Referer. Even when the personal information has been removed, the Referer header might indicate a private document's URI whose publication would be inappropriate.

> To quote from: the security Considerations section of the HTTP 1.1 protocol specification (http://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html#sec15)
>
> HTTP clients are often privy to large amounts of personal information (e.g. the user's name, location, mail address, passwords, encryption keys, etc.), and SHOULD be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources. We

very strongly recommend that a convenient interface be provided for the user to control dissemination of such information, and that designers and implementors be particularly careful in this area. History shows that errors in this area often create serious security and/or privacy problems and generate highly adverse publicity for the implementor's company.

The server acknowledges the input and then commences download of the web page data

```
  www.isoc.org.80         >         gih.example.com.1044:         .
2174176571:2174176571(0) ack 72886974 win 31740 (DF) (ttl 47, id
24463, len 40)
```

Here's the first two cycles of delivery of the data.

```
 www.isoc.org.80         >         gih.example.com.1044:          P
2174176571:2174177951(1380) ack 72886974 win 31740 (DF) (ttl 47,
id 24464, len 1420)
 gih.example.com.1044  >  www.isoc.org.80:  .  72886974:72886974(0)
ack 2174177951 win 64860 (DF) (ttl 128, id 320, len 40)
 www.isoc.org.80         >         gih.example.com.1044:          P
2174177951:2174179331(1380) ack 72886974 win 31740 (DF) (ttl 47,
id 24465, len 1420)
 www.isoc.org.80         >         gih.example.com.1044:          P
2174179331:2174180711(1380) ack 72886974 win 31740 (DF) (ttl 47,
id 24467, len 1420)
 gih.example.com.1044  >  www.isoc.org.80:  .  72886974:72886974(0)
ack 2174180711 win 64860 (DF) (ttl 128, id 323, len 40)
```

Is the object delivered by the server that original object that was placed there by the content originator? Servers are vulnerable to attack, and defacing a web site by substituting altered content in the place of the original is not uncommon. While some forms of content substitution are readily apparent to the user, other forms of substitution may be far more subtle in their intent, attempting to convince the user that the content is genuine and that the web transaction is trustworthy. Other forms of attack to the web transaction are more difficult to trace. If the user's request is passed through a web cache, then an attack on the cache can result in the delivery of substituted content to the user while the content originator's site has maintained its integrity.

So when you enter a URL and get back a page there's a lot of trust is happening. Probably too much trust.

If thats the case, then how can we improve the situation?

The basic toolset is all about authentication and validation, together with the ability to be able to conduct a private data exchange even across a public medium. Paradoxically, many of the basic tools are already available. But being available and being used are different concepts, and

There are effective cryptographic algorithms tools in widespread use. Whether its the Data Encryption Standard (DES), Triple DES, the International Data Encryption Algorithm (IDEA), The ADvanced Encryption Standard (AES), Blowfish, Twofish or any other block cipher, it is possible to

encrypt a data exchange between two parties to a level where there is reasonable confidence that the data will remain private.

Of course, as with many other secure systems, its not the encryption algorithm that is the vulnerable point, but the initial key exchange that sets up the communication using a shared secret, namely the encryption key. If this key exchange is compromised, then so is the integrity of the communication.

The common solution to this is the concept of private and public key pairs. While the mathematics behind this is well beyond the scope of this article, the essential characteristics of a public / private key pair is that anything encoded using the private key can only be decoded using the corresponding public key, and anything encoded using the public key and only be decoded using the private key. And, most importantly, knowledge of the public key will not allow you to compute the value of the private key. Now we have the elements of a more robust environment. If you know my public key you can encode some data using this public key and pass it to me. Only my private key can decode it, and, as long as I keep this private key a secret then only I can read your original data. And if you then encode this data using your private key, then I will need to decode the data first with your public key and then with my private key. At this point we have a mechanism to undertake a relatively secure form of key exchange, where each party can identify the other, and exchange messages that only the other party can read. But it hinges on being able to associate a public key with the remote party in a trustable manner. And here's where a trustable public key infrastructure comes into play.

Widespread use of a trustable public key infrastructure would assist in allowing users to validate that the content has been generated by the author. In a model of complementary public and private key pairs a web object can be signed using the object originator's private key. If the object's signature can be validated against the originator's public key then there is a strong indication that the content is genuine and has not been altered in any way.

Email can also benefit from widespread use of the same public key cryptography, where a message can be signed with the author's private key and the signed message can then be encrypted using a key that is encoded using the intended recipient's public key. Assuming that the keys have not been compromised, such a message can only be read by the intended recipient, and could only have been sent by the owner of the digital signature.

So can we trust this answer? No not really.

Imagine for a second if a public key infrastructure was in widespread use and we only accepted email that was digitally signed using keys that we trust. Imagine also that web objects were also signed in a similar fashion. In such a world how would a spammer hide their true identity? If spammers cannot hide behind anonymity, then would spam be as prevalent in such a network? Would various ecommerce applications enjoy greater acceptance if users were confidant that there was both privacy and authenticity associated with such transactions? What would be the point of subverting a web site or its content if the action were immediately visible as an unauthorized alteration? Not only would such a network mitigate some of the problems we see today, but its likely that we would find new services appearing that rely on a useful end-to-end trust model where each end can validate the identity of the other party and also validate the authenticity and privacy of the exchanged data.

We can go beyond the application level in this, an d look at how we can improve upon the trust model in the network itself and its associated service layer. DNSSEC allows a DNS server to ensure that only known remote agents can provide updates to the server, and that the updates are not altered by a third party on the fly. If DNSSEC were an integral part of the operation of the DNS it

would be a far more difficult proposition to attempt to insert incorrect data into the DNS. At the network level, providers commonly operate Internet networks with efficiency and unit cost in mind. In-band operation of routing protocols readily fit within such an operational model. The SNMP-based model of network management is also commonly operated as an in-band tool. The result is that the operational management of the network, the monitoring of the network and the payload carried by the network all share the same transport system. There is the risk that a payload packet can masquerade as a routing control packet, or an SNMP write transaction, and if the network does not detect this as an exception, the network can be disrupted, or even conceivably taken over by a third party. The trust model here is obviously not that the network operator trusts users not to generate such attacks. The appropriate model is that the network operator distrusts users, and takes appropriate precautions to ensure that a network element will only accept control and monitoring packets that truly originate from the network operator's internal control points.

It would be useful to have each registry-allocated address block be associated with a public key. This would allow all new routing requests to be signed by the private key associated with the original address allocation, which in turn would assist the ISP in determining the authenticity of the request before entering the address prefix into the Internet's routing system.

Its not that we can eliminate the need for some degree of trust from the Internet, nor that all potential security risks can be comprehensively addressed. But in using a set of basic security tools with our existing network applications we can make some significant improvements upon the current rather open trust model of the Internet. In so doing we make significant improvements in the level of trust users can reasonably place on the integrity of the Internet as a useful communications medium. And surely that's a good thing.

Further Reading

"**Network Security** PRIVATE Communication in a Public World"

Disclaimer

The author is a member of the Internet Architecture Board (IAB). The opinions expressed in this article are entirely those of the author, and are not necessarily shared by the IAB as a whole.

The above views do not represent the views of the Internet Society, nor do they represent the views of the author's employer, the Telstra Corporation. They were possibly the opinions of the author at the time of writing this article, but things always change, including the author's opinions!

About the Author

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for the past decade, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. Huston is currently the Chief Scientist in the Internet area for Telstra. He is also a member of the Internet Architecture Board, and is the Secretary of the APNIC Executive Committee. He was an inaugural Trustee of the Internet Society, and served as Secretary of the Board of Trustees from 1993 until 2001, with a term of service as chair of the Board of Trustees in 1999 – 2000. He is author of *The ISP Survival Guide*, ISBN 0-471-31499-4, *Internet Performance Survival Guide: QoS Strategies for Multiservice Networks*, ISBN 0471-378089, and coauthor of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, ISBN 0-471-24358-2, a collaboration with Paul Ferguson. All three books are published by John Wiley & Sons.

E-mail: gih@telstra.net