The Internet Protocol Journal

March 2012 Volume 15, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From the Editor 1

Hacking Internet Security 2

DANE..... 12

Twenty-Five Years Ago..... 24

Letter to the Editor 36

Fragments 37

You can download IPJ back issues and find subscription information at:
www.cisco.com/ipj

ISSN 1944-1134

From The Editor

Internet security continues to receive much attention both in the media and within the *Internet Engineering Task Force* (IETF) and similar organizations that develop technical solutions and standards. Last September, someone managed to break into a trusted *Certification Authority*'s system and subsequently produced numerous fake *digital certificates*, files that comprise part of the architecture for what is generally referred to as "browser security." In our first article, Geoff Huston describes what happened, the implications of this form of attack on the security of web-based services on the Internet, and what can be done to prevent similar attacks in the future.

In our second article, Richard Barnes describes the work of the *DNS-based Authentication of Named Entities* (DANE) working group in the IETF and explains how DANE, when deployed, can help prevent the sort of attack that is described in our first article.

This year I am celebrating 25 years in Internet technical publishing. Prior to launching *The Internet Protocol Journal* (IPJ), I was the editor of *ConneXions—The Interoperability Report,* published from 1987 until 1997 by Interop Company. With the generous support of *The Charles Babbage Institute* at the University of Minnesota, *ConneXions,* which was a paper-only publication, has been scanned and made available online. To mark the 25 combined years of *ConneXions* and IPJ, we asked Geoff Huston to examine the state of computer communications 25 years ago and give us his thoughts on where we have been and where we might be going in this rapidly developing technology landscape.

Please remember to check your subscription expiration date and take the necessary steps if you wish to continue receiving this journal. You will need your subscription ID and the e-mail address you used when you subscribed in order to access your record and renew online. Visit the IPJ website at `www.cisco.com/ipj` and click on the "Subscriber Services" link to get to the login page. The system will send you a URL that allows direct access to your record. If you no longer have access to the e-mail you used when you subscribed or you have forgotten your subscription ID, just send a message to `ipj@cisco.com` and we will make the necessary changes for you.

—Ole J. Jacobsen, Editor and Publisher
`ole@cisco.com`

Hacking Away at Internet Security

by Geoff Huston, APNIC

The front page story of the September 13, 2011, issue of the *International Herald Tribune* said it all: "Iranian activists feel the chill as hacker taps into e-mails." The news story relates how a hacker has "... sneaked into the computer systems of a security firm on the outskirts of Amsterdam" and then "... created credentials that could allow someone to spy on Internet connections that appeared to be secure." According to this news report, the incident punched a hole in an online security mechanism that is trusted by hundreds of millions of Internet users all over the network.

Other news stories took this hyperbole about digital crime and tapping into e-mail conversations on the Internet to new heights, such as *The Guardian's* report on September 5, 2011, which claimed that the "... DigiNotar SSL certificate hack amounts to cyberwar, says expert."[1]

If application-level security is so vulnerable to attack, then this incident surely calls into question the basic mechanisms of trust and security upon which the entire global Internet has been constructed. By implication it also calls into question the trustworthiness of services operated by the major global Internet brands such as Google and Facebook, as much as it raises doubts about the levels of vulnerability for the use of online services such as banking and commercial transactions.

Just how serious is this problem? Are we now at the end of civilization as we know it?

Well, hardly!

Is digital cryptography now broken? Has someone finally managed to devise a computationally viable algorithm to perform prime factorization of massively large numbers, which lies at the heart of much of the cryptography used in the Internet today?

I really don't think so. (At the very least, if someone has managed to achieve this goal, then that person is staying very quiet about it.).

Does this situation represent a systematic failure of security? Do we need to rethink the entire framework of cryptography and security in the Internet?

Not this time.

As far as I can tell, there has been no dramatic failure in the integrity of the digital technology used for security in the Internet today. Yes, some were surprised by this failure, including the Netherlands government, which uses certificates issued by the compromised certification authority, DigiNotar (`http://www.diginotar.com`) as part of its online service infrastructure. But the hacking incident was not based on a successful direct attack on the technology of cryptography by itself, and there is no reason to suppose that the strength of today's encryption algorithms is any weaker today than yesterday.

But in observing that the basic technology tools of the Internet security framework are still operating within acceptable bounds of integrity, and observing that this hacking attack did not create a gaping hole in our commitment to digital cryptography, what cannot be claimed is that the use of these cryptographic tools in today's Internet service environment is similarly trustworthy. The hacking attempt apparently was successful in so far as it provided the capability for third parties to impersonate trusted services and thereby capture users' private data, and evidently some people did indeed do precisely that, and that is not good at all.

Let's look a little more closely at this hacking episode and examine the way in which security is applied to the world of web browsing and the manner in which the vulnerabilities in this security framework were evidently exploited.

Securing a Connection

When I point my browser at my online banking service—or at any other secure website for that matter—a part of the browser navigation bar probably glows a reassuring green, and when I click it I get the message that I am connected to a website run by the Acme Banking corporation, and that my connection to this website has been encrypted to prevent eavesdropping. However, the website *certificate* was issued by some company that I have never even heard of. When I ask for more information, I am told the domain name, the company to whom the certificate for this domain name was issued, the identity of the certificate issuer, and the public key value. I am also reassuringly informed that the message I am viewing was encrypted before being transmitted over the Internet, and that this encryption makes it very difficult for unauthorized people to view information travelling between computers, and it is therefore very unlikely that anyone could read this page as it passes through the network. All very reassuring, and for the most part true, to the extent that we understand the strength of cryptographic algorithms in use today. The connection is using a *Transport-Layer Security* (TLS)[2] connection and the traffic is encrypted using a private session key that should be impenetrable to all potential eavesdroppers.

But that is not the entire truth, unfortunately.

It may well be that your conversation is secure against eavesdropping, but it is only as secure as the ability of the other party to keep its private key a secret. If the other side of the conversation were to openly broadcast the value of its private key, then the entire encryption exercise is somewhat useless. So, obviously, my local bank will go to great lengths to keep its private key value a secret, and I rely on its efforts in order to protect my conversations with the bank.

But even then it is not quite the full story.

Am I really talking to my bank? Or in more general terms, am I really talking to the party with whom I wanted to talk?

The critical weakness in this entire framework of security is that the binding of certificates and keys to *Domain Name System* (DNS) names is not an intrinsic part of the DNS itself. It is not an extension of *Domain Name System Security Extensions* (DNSSEC)[3, 4]. It has been implemented as an add-on module where third parties generate certificates that attest that someone has a particular domain name. Oddly enough, these *Certification Authorities* (CAs) may never have actually issued that particular domain name, because they are often disconnected from the DNS name registration business. Their business is a separate business activity where, after you have paid your money to a domain name registrar and secured your domain name, you then head to a domain name Certification Authority and pay them money (commonly they charge more money than the name registration itself) and receive a domain name certificate.

Certification Authorities

Who gets to be a Certification Authority? Who gets to say who has which domain name and what keys should be associated with that domain name?

Oddly enough the answer is, at a first level of approximation, just about anyone who wants to! I could issue a certificate to state that you have the domain name **`www.example.com`** and that your public key value is some number. The certificate I issue to that effect would not be much different from the certificates issued by everyone else. Yes, my name would be listed as the certificate issuer, but that is about all in terms of the difference between this certificate and the set of certificates you already trust through your browser.

So what is stopping everyone from being a Certification Authority? What is preventing this system from descending into a chaotic environment with thousands of certificate issuers?

For this situation the browser software folks (and other application developers of secure services) have developed a solution. In practice it requires a lot of effort, capability, diligence, and needless to say, some money, to convince a browser to add your Certification Authority public key to its list of trusted Certification Authorities.

You have to convince the browser developers that you are consistently diligent in ensuring that you issue certificates only to the "correct" holders of domain names and that you undertake certificate management practices to the specified level of integrity and trust. In other words, you have to demonstrate that you are trustworthy and perform your role with consistent integrity at all times. You then get listed with all the other trusted Certification Authorities in the browser, and users will implicitly trust the certificates you issue as part of the security framework of the Internet.

How many trusted Certification Authorities are there? How many entities have managed to convince browser manufacturers that they are eminently trustable people? If you are thinking that this role is a special one that only a very select and suitably *small* number of folks who merit such absolute levels of trust should undertake for the global Internet—maybe two or three such people—then, sadly, you are very much mistaken.

Look at your browser in the preferences area for your list of trusted Certification Authorities, and keep your finger near the scroll button, because you will have to scroll through numerous such entities. My browser contains around 80 such entities, including one government ("Japanese Government"), a PC manufacturer ("Dell Inc"), numerous telcos, and a few dedicated certificate issuers, including DigiNotar.

Do I know all these folks that I am meant to trust? Of course not! Can I tell if any of these organizations are issuing rogue certificates, deliberately—or far more likely—inadvertently? Of course not!

The structural weakness in this system is that a client does not know *which* Certification Authority—or even which duly delegated subordinate entity of a Certification Authority—was used to issue the "genuine" DNS certificate. When a client receives a certificate as part of the TLS initialization process, then as long as any one of the listed trusted Certification Authorities is able to validate the presented certificate, even if it is the "wrong" Certification Authority, then the client will proceed with the session with the assumption that the session is being set up with the genuine destination.

In other words the entire certification setup is only as strong—or as weak—as the weakest of the certification authorities. It really does not matter to the system as a whole if any single Certification Authority is "better" at its task than the others, because every certified domain name is protected only to the extent that the "weakest" or most vulnerable trusted Certification Authority is capable of resisting malicious attack and subversion of its function. Indeed, one could argue that there is scant motivation for any trusted Certification Authority to spend significantly more money to be "better" than the others, given that its clients are still as vulnerable as all the other clients of all the other Certification Authorities.

In other words, there is no overt motivation for market differentiation based on functional excellence, so all certificates are only as strong as the weakest of all the Certification Authorities. And therein lies the seed of this particular hacking episode.

The Hack

The hack itself now appears to have been just another instance of an online break-in to a web server. The web server in question was evidently running the service platform for DigiNotar, and the hacker was able to mint some 344 fraudulent certificates, where the subject of the certificate was valid, but the public key was created by the hacker. A full report of the hacking incident was published by Fox-IT[5].

To use these fraudulent certificates in an attack requires a little more than just minting fraudulent certificates. It requires traffic to be redirected to a rogue website that impersonates the webpage that is under attack. This redirection requires collusion with a service provider to redirect client traffic to the rogue site, or a second attack, this time on the Internet routing system, in order to perform the traffic redirection.

So minting the fraudulent certificates is just one part of the attack. Were these fake certificates used to lure victims to fake websites and eavesdrop on conversations between web servers and their clients? Let's look at the client's validation process to see if we can answer this question.

When starting a TLS session, the server presents the client with a certificate that contains the server public key. The client is expected to validate this certificate against the client's locally held set of public keys that are associated with trusted certification authorities. Here is the first vulnerability. The client is looking for any locally cached trusted key to validate this certificate. The client is not looking as to whether a particular public key validates this certificate. Let's say that I have a valid certificate issued by the Trusted Certification Authority Inc. for my domain name, **`www.example.com`**. Let's also say that the server belonging to another Certification Authority, Acme Inc, is compromised, and a fake certificate is minted. If a user is misdirected to a fake instance of **`www.example.com`** and the bad server passes the client this fake certificate, the client will accept this fake certificate as valid because the client has no *presumptive* knowledge that the only key that should validate a certificate for **`www.example.com`** belongs to the Trusted Certification Authority Inc. When the key belonging to Acme Inc validates this certificate and ACME is a trusted entity according to my browser, then that is good enough to proceed.

Actually that is not the full story. What if I wanted to cancel a certificate? How do certificates get removed from the system and how do clients know to discard them as invalid?

A diligent client (and one who may need to check a box in the browser preference pane to include this function) uses a second test for validity of a presented certificate, namely the *Online Certificate Status Protocol* (OCSP)[6]. Clients use this protocol to see if an issued certificate has been subsequently revoked. So after the certificate has been validated against the locally held public key, a diligent application will then establish a secure connection to the certification authority OCSP server and query the status of the certificate.

This secure connection allows for prompt removal of fraudulent certificates from circulation. It assumes of course that clients use OCSP diligently and that the Certification Authority OCSP server has not also been compromised in an attack, but in an imperfect world this step constitutes at least another measure of relative defence.

The OCSP server logs can also provide an indication of whether the fraudulent certificates have been used by impersonating servers, because if the certificate was presented to the client and the client passed it to an OCSP server for validation, then there is a record of use of the certificate. The Fox-IT report contains an interesting graphic that shows the geolocation of the source addresses of clients who passed a bad `*.google.com` certificate to OCSP for validation. The source addresses have a strong correlation to a national geolocation of Iran.

Obviously this attack requires some considerable sophistication and capability, hence the suspicion that the attack may have had some form of state or quasi-state sponsorship, and hence the headlines from *The Guardian,* quoted at the start of this article, that described this attack as an incident of cyberwarfare of one form or another. Whether this incident was a cyber attack launched by one nation state upon another, or whether this was an attack by a national agency on its own citizens is not completely clear, but the available evidence points strongly to the latter supposition.

Plugging the Hole?

This incident is not the first such incident that has created a hole in the security framework of the Internet, and it is my confident guess that it will not be the last. It is also a reasonable guess that the evolution of the sophistication and capability that lie behind these attacks points to a level of resourcing that leads some to the view that various state-sponsored entities may be getting involved in these activities in one way or another.

Can we fix this?

It seems to me that the critical weakness that was exploited here was the level of disconnection between domain name registration and certificate issuance. The holders of the domain names were unaware that fraudulent certificates had been minted and were being presented to users as if they were the real thing. And the users had no additional way of checking the validity of the certificate by referring back to information contained in the DNS that was placed there by the domain name holder.

The end user was unable to refine the search for a trusted Certification Authority that would validate the presented certificate from all locally cached trusted Certification Authorities to the one certification authority that was actually used by the domain name holder to certify the public key value. So is it possible to communicate this additional information to the user in a reliable and robust manner?

The last few years have seen the effort to secure the DNS gather momentum. The root of the DNS is now DNSSEC-signed, and attention is now being focused on extending the interlocking signature chains downward through the DNS hierarchy. The objective is a domain name framework where the end client can validate that the results returned from a DNS query contain authentic information that was entered into the DNS by the delegated authority for that particular DNS zone.

What if we were able to place certificates—or references to certificates—into the DNS and protect them with DNSSEC? The *DNS-based Authentication of Named Entities* (DANE) Working Group of the IETF[0, 7] is considering this area of study. They are considering numerous scenarios at present, and the one of interest here does not replace the framework of Certification Authorities and domain name certificates, but it adds another phase of verification of the presented certificate.

The "Use Cases"[8] document from the DANE working group illustrates the proposed approach. I will quote a few paragraphs from this document. The first paragraph describes the form of attack that was perpetrated in June and July this year on the DigiNotar CA. It is not clear to me if the text predates this attack or not, but they are closely aligned in time:

> "Today, an attacker can successfully authenticate as a given application service domain if he can obtain a 'mis-issued' certificate from one of the widely-used CAs—a certificate containing the victim application service's domain name and a public key whose corresponding private key is held by the attacker. If the attacker can additionally insert himself as a man in the middle between a client and server (for example, through DNS cache poisoning of an A or AAAA record), then the attacker can convince the client that a server of the attacker's choice legitimately represents the victim's application service."[8]

So how can DNSSEC help here?

> "With the advent of DNSSEC [RFC 4033], it is now possible for DNS name resolution to provide its information securely, in the sense that clients can verify that DNS information was provided by the domain holder and not tampered with in transit.

> The goal of technologies for *DNS-based Authentication of Named Entities* (DANE) is to use the DNS and DNSSEC to provide additional information about the cryptographic credentials associated with a domain, so that clients can use this information to increase the level of assurance they receive from the TLS handshake process.
>
> This document describes a set of use cases that capture specific goals for using the DNS in this way, and a set of requirements that the ultimate DANE mechanism should satisfy. Finally, it should be noted that although this document will frequently use HTTPS as an example application service, DANE is intended to apply equally to all applications that make use of TLS to connect to application services named by domain names."[8]

Does DANE represent a comprehensive solution to this security vulnerability?

I would hesitate to be that definitive. As usual with many aspects of security, the objective of the defender is to expend a smaller amount of effort in order to force an attack to spend a far larger amount of effort. From this perspective, the DANE approach appears to offer significant promise because it interlocks numerous security measures and forces a potential attacker to compromise numerous independent systems simultaneously. Within the DANE framework the attacker cannot attack any certification authority, but must compromise a particular certification authority, and the attacker must also attack DNSSEC and compromise the information contained in signed DNS responses for that domain in order to reproduce the effects of the attack described here. This scenario seems to fit the requirement of a small amount of additional defensive effort by the server and the client, creating a significantly larger challenge to the attacker.

But many preconditions must be met here for this approach to be effective:

- DNSSEC needs to be ubiquitously deployed and maintained.
- Issued DNS certificates need to be published in the secure DNS zone using the DANE framework.
- Client DNS resolvers need not only to be DNSSEC-aware, but also to enforce DNSSEC outcomes.
- Applications, including browsers, need to validate the certificate that is being used to form the TLS connection against the information provided by a validated DNS response for the DANE credentials for that DNS zone.

It is probably not perfect, but it is a large step forward along a path of providing more effective security in the Internet.

Unfortunately, this solution does not constitute an instant solution ready for widespread use today—or even tomorrow. We could possibly see this solution in widespread use in a couple of years, but, sadly, it is more likely that securing the DNS for use in the Internet will not receive adequate levels of attention and associated financial resourcing in the coming years. It may take upward of 5 years before we see ubiquitous adoption of DNSSEC and any significant levels of its use by a DANE framework for certificates in the DNS. Until then there is the somewhat worrisome prospect of little change in the framework of Internet security from that used today, and the equally concerning prospect that this particular hacking event will not be the last.

Acknowledgement

I am indebted to Olaf Kolkman of NLnet Labs for a stimulating conversation about this attack and the implications for securing the Internet. NLnet Labs is one of a small number of innovative and highly productive research groups that has developed considerable levels of expertise in this area of security and the DNS.[9]

Postscript

When you lose that essential element of trust, your continued existence as a trusted Certification Authority is evidently a very limited one. On Tuesday September 20, 2011, the Dutch company DigiNotar was officially declared bankrupt in a Haarlem court.

Disclaimer

The views of this article do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

References

[0] Richard L. Barnes, "Let the Names Speak for Themselves: Improving Domain Name Authentication with DNSSEC and DANE," *The Internet Protocol Journal,* Volume 15, No. 2, March 2012.

[1] **http://www.guardian.co.uk/technology/2011/sep/05/diginotar-certificate-hack-cyberwar**

[2] William Stallings, "SSL: Foundation for Web Security," *The Internet Protocol Journal,* Volume 1, No. 1, June 1998.

[3] Miek Gieben, "DNSSEC: The Protocol, Deployment, and a Bit of Development," *The Internet Protocol Journal,* Volume 7, No. 2, June 2004.

[4] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, "DNS Security Introduction and Requirements," RFC 4033, March 2005.

[5] Fox IT, "DigiNotar Certificate Authority breach, 'Operation Black Tulip,'"
http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2011/09/05/diginotar-public-report-version-1/rapport-fox-it-operation-black-tulip-v1-0.pdf

[6] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP," RFC 2560, June 1999.

[7] **http://datatracker.ietf.org/wg/dane/**

[8] Richard Barnes, "Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)," RFC 6394, October 2011.

[9] **http://nlnetlabs.nl/**

[10] On March 26, 2012, at IETF 83 in Paris, France, a Technical Session with the title "Implementation Challenges with Browser Security" was held. The following presentations were given:

- Hannes Tschofenig: "Introduction"
- Eric Rescorla: "How do we get to TLS Everywhere?"
- Tom Lowenthal: "Cryptography Infrastructure"
- Chris Weber: "When Good Standards Go Bad"
- Ian Fette: "Lessons Learned from WebSockets (RFC 6455)"
- Jeff Hodges: "It's Not the End of the World"

All of these presentations are available from:
https://datatracker.ietf.org/meeting/83/materials.html

GEOFF HUSTON B.Sc., M.Sc., is the Chief Scientist at *Asia Pacific Network Information Centre* (APNIC), the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of numerous Internet-related books, was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001. E-mail: **gih@apnic.net**

Let the Names Speak for Themselves: Improving Domain Name Authentication with DNSSEC and DANE

by Richard L. Barnes, BBN Technologies

Authentication of domain names is a fundamental function for Internet security. In order for applications to protect information from unauthorized disclosure, they need to make sure that the entity on the far end of a secure connection actually represents the domain that the user intended to connect to. For many years, authentication of domain names has been accomplished by having third-party *Certification Authorities* attest to which entities could represent a domain name. This system of external authorities, however, has recently come under heavy attack, and there have been several high-profile compromises[0]. The *Domain Name System Security Extensions* (DNSSEC) offer an alternative channel for distributing secure information about domain names, through the *Domain Name System* (DNS) itself. The *DNS-based Authentication of Named Entities* (DANE) working group in the *Internet Engineering Task Force* (IETF) has developed a new type of DNS record that allows a domain itself to sign statements about which entities are authorized to represent it. End users' applications can use these records either to augment the existing system of Certification Authorities or to create a new chain of trust, rooted in the DNS.

Authentication

Without authentication, other security services are moot. There is little point in Alice's encrypting information en route to Bob if she has not first verified that she is talking to Bob and not an attacker Eve. In the context of Internet applications, authentication is about ensuring that users know whom they are talking to, and in most cases, that "whom," is represented by a domain name. For example, in the *Hypertext Transfer Protocol* (HTTP), the "authority" section of a *Uniform Resource Identifier* (URI) indicates the domain name of the server that will fulfill requests for that URI. So when an HTTP user agent starts a TCP connection to a remote server, it needs to verify that the server is actually authorized to represent that domain name[1].

The most common security protocol used by Internet applications is the *Transport Layer Security* (TLS) protocol[2]. TLS provides a layer above TCP that facilitates authentication of the remote side of the connection as well as encryption and integrity protection for data. TLS underlies *Secure HTTP* (HTTPS) and secure e-mail[1, 3, 4], and provides hop-by-hop security in real-time multimedia and instant-messaging protocols[5, 6]. In all of these applications, the server that the user ultimately wants to connect to is identified by a DNS domain name[7, 8]. A user might enter `https://example.com` into a web browser or send an e-mail to `alice@example.com`.

One of the main purposes of using TLS in these cases is thus to assure the user that the entity on the other end of the connection actually represents **`example.com`**; in other words, to authenticate the server as a legitimate representative of the domain name. Note that these comments apply to *Datagram Transport Layer Security* (DTLS) as well, because it provides the same functions as TLS for *User Datagram Protocol* (UDP) packet flows[9].

Today, a server asserts its right to represent a domain by presenting a *Public Key Infrastructure* (PKIX) digital certificate containing that domain[8, 10]. A certificate is an attestation by a Certification Authority of a binding between a public key and a name—the entity holding the corresponding private key is authorized to represent that name. TLS ensures that only the holder of a given private key can read the encrypted data; the certificate ensures that the holder of the key represents the desired name.

Current TLS-based applications maintain a list of Certification Authorities whose certificates they will accept. Unfortunately, over time, these lists have grown very long, with major web browsers trusting nearly 200 Certification Authorities, representing a diverse range of organizations. Because any of these Certification Authorities can vouch for any domain name, a long list creates many points of vulnerability; a compromise at any point allows the attacker to issue certificates for any domain. Several recent attacks have taken advantage of this fact by targeting smaller Certification Authorities as a way to obtain certificates for major domains. For example, an attack through DigiNotar against Google is discussed in this issue[0].

DNSSEC offers an alternative to Certification Authorities. In the DNSSEC system, each domain holder can act as an authority for subordinate domains. The IETF DANE working group has developed a DNS record format for "certificate associations," so that domain holders can sign statements about which certificates can be used to authenticate as that domain. In effect, this scenario allows a domain to speak for itself, instead of through a third-party Certification Authority. DANE associations can be used either as a check on the current model (for example, to limit which Certification Authorities may vouch for a domain) or as an alternative trust path, rooting trust in a DNSSEC authority instead of a Certification Authority. Work on the protocol document is drawing to a close, and several prototype implementations are already in progress.

Background: PKIX and DNSSEC

At one level, the choice of which authentication technology to use is a choice of authorities and scoping. As mentioned previously, authentication is fundamental for security, but it is also very hard to accomplish scalably. For example, a web browser needs to be able to authenticate any website the user chooses to visit. It would clearly not work for each browser vendor to send a human representative to meet every website owner in order to find out what public key should be used for that website.

So instead of relying on having preestablished relationships with every entity we want to authenticate, we rely on centralized authorities to do identity checking. The authorities then create credentials that anyone else can check, so that if the credential is valid and you believe the authority is trustworthy, then the entity holding the credential has the indicated identity.

In a technical sense, an entity holds a credential if it holds the private key corresponding to the public key in the credential. The credential encodes a *binding* between the public key and the identity, asserted by the authority.

Authority is of course not a purely digital concept. If we want to know a person's name in real life we do not just ask them directly, because the person could lie. Instead we look to a credential issued by an authority, such as a driver's license or birth certificate. So the technology question here is how to manage authorities, and how to encode these credentials.

The IETF has defined two major cryptographic authority systems: PKIX, based on digital certificates[10]; and DNSSEC, based on the DNS[11]. Both of these systems allow authorities to associate public keys with identities, and both arrange these authorities hierarchically.

The hierarchy is important because it allows a *relying party* (someone who is verifying identities) to choose whom to trust. In these hierarchical systems, an authority's identity can itself be attested by a credential issued by another authority. When a relying party wants to verify a credential issued by an authority A, he then has to verify that A's credential is valid (under an authority B), and so on until he reaches an authority that he trusts. This sequence of credentials constitutes a logical path through the hierarchy, known as a "certification path" in PKIX terminology (Figure 1).

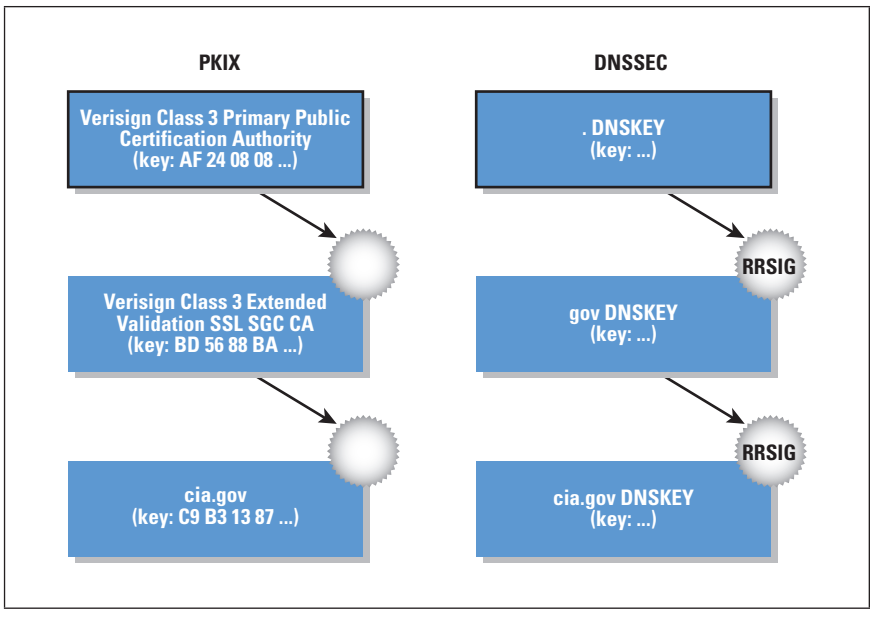


Figure 1: PKIX and DNSSEC Trust Hierarchies

In order to be useful as a given relying party to authenticate someone, a certification path has to end in a *trust anchor,* that is, an authority that the relying party trusts to make assertions. In the DNSSEC context, relying parties can in principle have only one trust anchor, namely the DNS root, although alternatives to the root have been proposed[12]. The PKIX system, on the other hand, does not represent a single globally consistent hierarchy, so in order to be able to validate many certificates, relying parties often have to choose many trust anchors.

Crossing the Streams

Current TLS-based applications rely on PKIX for authentication of domain names, which has facilitated fairly broad deployment, but also created some vulnerabilities. PKIX is based on a very general digital certificate system called X.509, and because of this generality, it has no inherent binding to the DNS. This situation creates two problems when it comes to authenticating domain names.

First, unlike the DNS, which has a single global root, there is no single authority under which all PKIX certificates can be verified. Indeed, there is an open marketplace of authorities, where each entity can choose which authority will sign its certificate, leaving relying parties with a choice: Either they must trust every authority that has signed a certificate for an entity it wants to authenticate, or they will be unable to validate the identities of some entities. In general, current software has preferred the former approach of trusting many authorities, to the extent that modern browsers and operating systems will trust up to 200 authorities by default. Users can add to this list, for example, using the "Accept this certificate?" dialogs in their browsers, but it can be very difficult to remove trust anchors from the default list[13].

Second, PKIX authorities today are not constrained in the scope, so they can issue credentials for any name—even those for whom they have no real information (in contrast to the DNS—where each zone can vouch only for sub-domains; only the root can act with impunity). Conversely, there is no real way for a relying party to know what authority should be vouching for a site, so if a rogue authority were to issue a certificate to an unauthorized party, relying parties would have no way to detect it.

Given these vulnerabilities, any of the many authorities trusted within the PKIX system can attack any domain by issuing a false certificate from that domain. This false certificate can then be used to masquerade as the victim domain, for example, to perform a man-in-the-middle attack. Note that the authority itself is not necessarily the bad actor in this attack—it could be an external attacker that can obtain illicit access to the systems that issue certificates. The risks of having broadly trusted Certificatation Authorities have recently become clear, because attackers were able to break into two small Certification Authorities and create fraudulent certificates for Google and Facebook, among others[14, 15].

The goal of DANE is to address some of the vulnerabilities of the current PKIX ecosystem by allowing DNSSEC—to "cross the streams" to allow domains to publish information secured with DNSSEC that can add additional security to PKIX certificates used for TLS. For example, a domain might use DANE to inform relying parties of which authorities can be trusted, as illustrated in Figure 2.

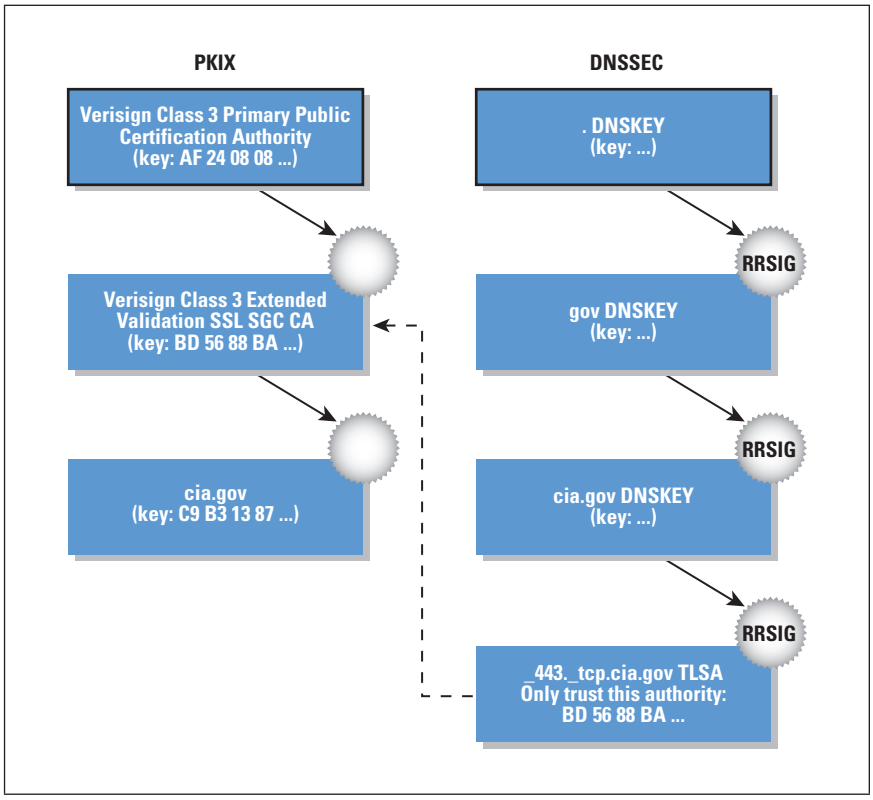


Figure 2: Using a DANE TLS Associations (TLSA) Record to Indicate Which PKIX Authority Should Be Trusted

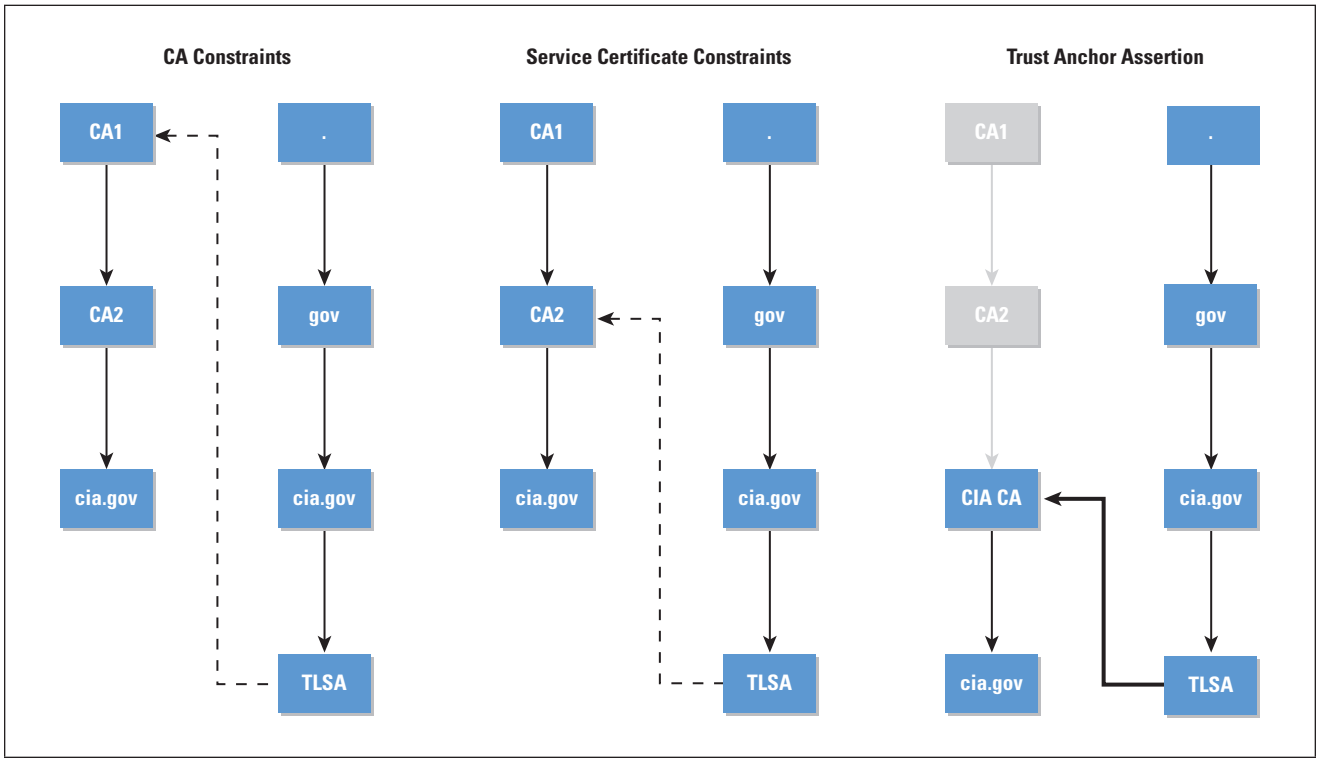
DANE Records

If the goal of DANE is to allow domain operators to make statements about how clients should judge TLS certificates for their domains, then what sorts of statements should DANE allow them to make? The DANE use cases document[16] lays out three major types of statements (Figure 3):

1. *CA Constraints:* The client should accept only certificates issued under a specific Certificatation Authority.
2. *Service Certificate Constraints:* The client should accept only a specific certificate.
3. *Trust Anchor Assertion:* The client should use a domain-provided trust anchor to validate certificates for that domain.

All three of these statements can be viewed as constraining the scope of trust anchors. The first two types limit the scope of existing trust anchors, whereas the third provides the client with a new trust anchor (still within a limited scope). More on these anchors in a moment.

Figure 3: DANE Use Cases



The current draft DANE protocol defines a DNS Resource Record type TLSA for describing *"TLS Associations"*—statements about what certificates are "associated" to a domain[17]. Each TLSA record has three basic fields:

- *Usage:* Which type of statement this record is making
- *Selector/Matching:* How a TLS certificate chain should be matched against this record (for example, by exact match, by public key, or by SHA-1 digest)
- *Certificate for Association:* The actual data against which the TLS certificate chain should be matched

These records are stored under the target domain with a prefix that indicates the transport and port number for the TLS server. So for example, if Alice runs a secure web service at **`example.com`** and wants to tell clients that they should accept only certificates from the Charlie's CA, she could provision a TLSA record under **`_443._tcp.example.com`** with the following contents:

- *Usage:* CA constraint
- *Selector/Matching:* SHA-1 digest
- *Certificate for Association:* SHA-1 digest of Charlie's certificate

When a client Bob wants to connect to **`https://example.com`**, he can find these TLSA records and apply Alice's constraints when he validates the server certificate.

Adding Constraints to PKIX

The major objective of the CA constraints and service certificate constraints is to guard against "mis-issue" of certificates. A certificate is "mis-issued" when a CA issues a certificate to an entity that does not actually represent the domain name in the certificate. Mis-issue can come about in many ways, including through malicious Certification Authorities, compromised Certification Authorities (as in the Comodo and DigiNotar example discussed previously), or Certification Authorities that are simply misled as to the attacker's identity through fraud or other means. Today, mis-issue can be difficult to detect, because there is no standard way for clients to figure out which Certification Authorities are supposed to be issuing certificates for a domain. When an attacker issued false certificates for the Google Gmail service under the DigiNotar Certification Authority, it was noticed only because a vigilant user posted to a Gmail help forum.[18]

By contrast, domain operators know exactly which Certification Authorities they have requested certificates from, and, of course, which specific certificates they have received. With DANE, the domain operator can convey this information to the client. For example, to guard against the DigiNotar attack, Google could have provisioned a TLSA record expressing a Certification Authority constraint with its real Certification Authority (which is not DigiNotar) or a certificate constraint with its actual certificate. Then DANE-aware clients would have been able to immediately see that the DigiNotar certificates were improperly issued and possibly indicative of a man-in-the-middle attack.

Empowering Domain Operators

According to data from the EFF SSL Observatory, which scans the whole IPv4 address space for HTTPS servers and collects their certificates, around 48 percent of all HTTPS servers present self-signed certificates[19]. An unknown number of other servers present certificates issued under Certification Authorities that are not in the major default trust anchor lists. For example, the United States Air Force web portal uses a certificate issued under a Department of Defense Certification Authority that is not trusted by Firefox[20]. In the current environment, most clients cannot authenticate these servers at all; they have to rely on users manually checking certificates, hopefully with some out-of-band information. As a result, these servers and their users are highly vulnerable to man-in-the-middle attacks against their supposedly secure sessions.

DANE Trust Anchor Assertions enable the operators of a domain to advertise a new trust anchor, under which certificates for that domain will be issued. Using these records, clients can dynamically discover what trust anchors they should accept for a given domain, instead of relying on a static list provided by a browser or operating system.

It may seem odd to talk about a domain supplying a client with trust anchors, because trust anchor provisioning is typically a very sensitive activity. If an attacker is able to install a trust anchor into a victim's trust anchor store, then the attacker can masquerade under any name he wants by issuing certificates under that name. The PKIX working group even defined a whole protocol for managing trust anchors[21].

DANE ensures that this trust anchor provisioning is secure by applying scoping and verifying that scoping using DNSSEC. DANE trust anchor assertions are scoped to a particular domain name, so even if an attacker can introduce a false trust anchor, he can use it to spoof only a single name. Furthermore, trust anchor assertions must be DNSSEC-signed, so clients can verify that the entity providing the trust anchor represents the domain in question. Ultimately, the client still has to have a list of trust anchors configured—but they are DNSSEC trust anchors instead of PKIX trust anchors.

Of course, in principle, a client needs only one trust anchor for DNSSEC, the root zone trust anchor. Because control of the DNS root does not change very often, it makes sense for this trust anchor to be statically configured!

The ability of a domain operator to explicitly indicate a trust anchor for a domain is obviously very powerful. It may be tempting to ask whether this case is really the only use case that DANE needs, that is, whether the constraint cases mentioned previously are needed at all. The answer is that the constraint cases are useful as a way to fold in PKIX validation with external Certification Authorities in addition to domain-asserted trust anchors. Most obviously, this feature is useful in transition, when not all clients will be DANE-aware. But even in the longer term, it is possible that Certification Authorities will be able to provide added value over DANE. For example, while DANE is made to bind certificates to domain names, Certification Authorities can vouch for bindings of certificates to other things, such as the legal identity and physical location attested in Extended Validation certificates[22].

Transition Challenges

As described previously, DANE offers some valuable new security properties for TLS authentication. But as with most IETF technologies—especially security technologies—there are some challenges to be overcome and some new potential pitfalls.

The most significant constraint for DANE deployment is DNSSEC deployment. On the server side, this problem is not a significant one because DNSSEC support is spreading fairly rapidly. On the client side, it may be more difficult. Although there are DNS libraries with robust DNSSEC support, many of the major DNS *Application Programming Interfaces* (APIs) that applications use do not provide any information about the DNSSEC status of the results returned.

So in order to implement DANE, application developers may have to re-factor their DNS support in addition to querying for some new record types. If more sites come to rely on DANE, then this process could also draw increasing attention to the various types of intermediaries that cause DNSSEC breakage (for example, home gateways that set DNS flags improperly).

Adding DNSSEC to the TLS connection process can also add significant latency to the TLS connection process. In addition to completing the TLS handshake and certificate validation, the client has to wait for several DNS round trips and then validate the chain of DNSSEC signatures. These combined delays can add up to multiple seconds of latency in connection establishment. Especially for real-time protocols such as HTTPS, *Session Initiation Protocol* (SIP), or *Extensible Messaging and Presence Protocol* (XMPP), such delay is clearly undesirable.

One mechanism proposed to mitigate these delays is to have the server pre-fetch all of the relevant DNSSEC records, namely all of the DS, DNSKEY, and RRSIG records chaining back to the root[27]. Then the server can provide a serialized version of the DNSSEC records in the TLS handshake, saving the client the latency of the required DNS queries. The details of this mechanism, however, are still being worked out among the DANE, TLS, and PKIX working groups[23]. A prototype version is now available in the Google Chrome web browser[24].

Security Considerations

From a security perspective, the major effect of DANE is the new role that DNS operators will play in securing Internet applications. Although DNSSEC has always meant that DNS operators would have more security functions, DANE deployment will give them an explicit effect on application security, acting as arbiters of who can authenticate under a given name in TLS. Especially if services use trust anchor assertions, DNS operators will play an analogous role to the one Certification Authorities play today—a compromise in a DNS operator will allow an attacker to masquerade as a victim domain (albeit for a more limited set of domains because of DANE constraints on names). So DNS operators are likely to inherit many of the security troubles that Certification Authorities experience today and will need to strengthen their security posture accordingly.

Another more subtle risk arises from the fact that the operator of a DNS zone is not always the same as the entity that is authorized to control the contents of the zone, which we will call the "domain holder." We used the phrase "domain operator" previously because DNSSEC protects DNS information only between the operator's name server and the client—it does not say that what is provisioned in the name server is authorized by the domain holder.

When a domain is operated by a third party, that third party is a point of vulnerability between the client and the holder of the domain. If the domain operator provides false DANE information through malice or compromise, then a client will not be able to distinguish it from genuine DANE information. To some extent, this risk is not really new; because many current Certification Authorities authenticate requests for domain certificates based on information that is under the control of the domain operator, domain operators can already influence the credentialing process. With DANE, however, the vulnerability is much easier to exploit, for example, because the DNS operator does not have to trick a third party. This vulnerability is also fundamental to protocols that rely on DNSSEC for security, and the implications for DANE are discussed in detail in the DANE use cases document[16]. The main mitigation is simply increased care on the part of domain holders to ensure that domain operators are not behaving badly.

Conclusions

For many years now, Internet applications have relied on assertions by third-party PKIX Certification Authorities to ensure that a server holding a particular private key was authorized to represent a domain. The promise of DANE is a more direct interaction between clients and the domains they interact with, secured by DNSSEC. In the short run, DANE can be deployed as an adjunct to the current system of certificates and authorities, adding constraints to better protect domains. In the long run, DANE will also allow domain operators to vouch for their own names.

The transition and security problems that face DANE are largely the growing pains of DNSSEC. It is not that DANE is causing these problems itself; rather, the problems arise because DANE is the first real application of DNSSEC that is expected to be widely deployed. So although it may be difficult to mitigate some of the security problems that DANE raises, and to enable more robust DNSSEC support in applications and gateways, these changes will ultimately make it simpler for applications to use DNSSEC for other purposes.

The DANE working group is making consistent progress on its deliverables, and there are already some prototype deployment tools. Their use cases document has been published as RFC 6394[16], and the corresponding document defining the TLSA record type is starting to mature[17]. As of this writing, it is in Working Group Last Call. On the client side, a variant of DANE has already been implemented in Google Chrome; on the server side, prototype tools are available to generate DANE records and to generate "DNSSEC-stapled" certificates based on DANE records[24, 25]. There is also an early-stage command-line tool for generating and verifying TLSA records[26].

References

[0] Geoff Huston, "Hacking Away at Internet Security," *The Internet Protocol Journal,* Volume 15, No. 1, March 2012.

[1] Eric Rescorla, "HTTP Over TLS," RFC 2818, May 2000.

[2] Tim Dierks and Eric Rescorla, Editors, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, August 2008.

[3] Chris Newman, "Using TLS with IMAP, POP3 and ACAP," RFC 2595, June 1999.

[4] Paul Hoffman, "SMTP Service Extension for Secure SMTP over Transport Layer Security," RFC 3207, February 2002.

[5] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[6] Peter Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 6120, March 2011.

[7] Paul Mockapetris, "Domain Names – Concepts and Facilities," RFC 1034, November 1987.

[8] Peter Saint-Andre and Jeff Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)," RFC 6125, March 2011

[9] Eric Rescorla and Nagendra Modadugu, "Datagram Transport Layer Security Version 1.2," RFC 6347, January 2012.

[10] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and Tim Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, May 2008.

[11] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, "DNS Security Introduction and Requirements," RFC 4033, March 2005.

[12] `https://dlv.isc.org/`

[13] `http://arstechnica.com/apple/news/2011/09/safari-users-still-susceptible-to-attacks-using-fake-diginotar-certs.ars`

[14] http://blogs.comodo.com/it-security/data-security/the-recent-ra-compromise/

[15] http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/

[16] Richard Barnes, "Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)," RFC 6394, October 2011.

[17] Paul Hoffman and Jakob Schlyter, "Using Secure DNS to Associate Certificates with Domain Names for TLS," Internet Draft, work in progress, draft-ietf-dane-protocol-16, February 2012.

[18] http://www.google.co.uk/support/forum/p/gmail/thread?tid=2da6158b094b225a&hl=en

[19] http://www.eff.org/observatory

[20] https://www.my.af.mil/

[21] Russ Housley, Sam Ashmore, and Carl Wallace, "Trust Anchor Management Protocol (TAMP)," RFC 5934, August 2010.

[22] http://cabforum.org/Guidelines_v1_2.pdf

[23] Adam Langley, "Serializing DNS Records with DNSSEC Authentication," Internet Draft, work in progress, July 2011, draft-agl-dane-serializechain.

[24] http://www.imperialviolet.org/2011/06/16/dnssec-chrome.html

[25] https://dane.xelerance.com/

[26] https://github.com/pieterlexis/swede

[27] Wikipedia, "List of DNS record types," http://en.wikipedia.org/wiki/List_of_DNS_record_types

RICHARD BARNES has been with BBN Technologies since 2005. Richard is a member of BBN's Internet standards security team. In that role, he currently leads BBN's IETF standards efforts in the areas of geolocation, presence, and emergency calling. He is chair of the IETF GEOPRIV working group and a member of the IETF Security Area Directorate (SECDIR), and he is one of the program chairs of the Emergency Services Workshop. Prior to joining BBN, he was a student at the University of Virginia (United States), from which he received a B.A. and M.S. in Mathematics, with research focused on biologically based neural networks, quantum informatics, and network security. E-mail: rbarnes@bbn.com

A Retrospective: Twenty-Five Years Ago

by Geoff Huston, APNIC

The Information Technology business is one that rarely pauses for breath. Gordon Moore noted in 1965 that the number of components in integrated circuits had doubled every year from 1958 to 1965, and confidently predicted that this doubling would continue "for at least 10 years." This feature has been a continuing feature of the silicon industry for the past 50 years now, and its constancy has transformed this prediction into *Moore's Law.* The implications of this constant impetus for innovation in this industry have resulted in an industry that is incapable of remaining in stasis, and what we have instead is an industry that completely reinvents itself in cycles as short as a decade.

Looking back over the past 25 years, we have traversed an enormous distance in terms of technical capability. The leading silicon innovations of the late 1980s were in the Intel 80486 chip, which contained 1 million transistors on a single silicon chip with a clock speed of 50 MHz, and a similarly capable Motorola 68040 processor. Twenty-five years later the state of the art is a multicore processor chip that contains just under 3 billion individual transistors and clock speeds approaching 4 GHz. And where has all that processing power gone? In the same period we have managed to build extremely sophisticated programmed environments that have produced such products as Apple's *Siri* iPhone application, which combines voice recognition with a powerful information manipulation system, and we have packaged all of this computing capability into a device that fits comfortably in your pocket with room to spare!

Given that the last 25 years in IT has been so active, to look back over this period and contemplate all that has happened is a daunting task, and I am pretty sure that any effort to identify the innovative highlights in that period would necessarily be highly idiosyncratic. So instead of trying to plot the entire story that took us from then to now, I would like instead just to look at "then." In this article, to celebrate 25 combined years of *The Internet Protocol Journal* (IPJ)[2, 3] and its predecessor *ConneXions—The Interoperability Report*[0], I would like to look at the networking environment of the late 1980s and see what, if anything, was around then that was formative in shaping what we are doing today, and how it might influence our tomorrow.

The Computing Landscape of the Late 1980s

The computing environment of the late 1980s now seems to be quite an alien environment. Obviously there were no pocket-sized computers then. Indeed there were no pocket-sized mobile phones then. (I recall a visit from a salesman at the time who sported the very latest in mobile telephony—a radio setup that was the size of a briefcase!)

In 1987 the IT world was still fixated with the mainframe computer, which was basking in its last couple of years of viability in the market. IBM enjoyed the dominant position in this marketplace, and *Digital Equipment Corporation* (DEC) was competing with IBM with its VAX/VMS systems. These systems were intended to take the place of the earlier DEC-10 architectures, as well as offering an upgrade path for the hugely successful PDP-11 minicomputer line. The typical architecture of the computing environment was still highly centralized, with a large multiuser system at its core, and an attendant network or peripheral devices. These peripheral devices were traditionally video terminals, which were a simple ASCII keyboard and screen, and the interaction with the mainframe was through simple serial line character-based protocols.

Although it may not have been universally accepted at the time, this period at the end of the 1980s marked the end of the custom-designed mainframe environment, where large-scale computer systems were designed as a set of component subsystems, placed into a rack of some sort and interconnected through a bus or blackplane. Like many other human efforts, as far as the mainframe computer sector was concerned its final achievements were its greatest.

While the mainframe sector was inexorably winding down, at the other end of the market things were moving very quickly. The Zylogics Z80 processor of the mid-1970s had been displaced by the Intel 8080 chip, which evolved rapidly into 16-bit, then 32-bit processor versions. By 1987 the latest chip was the Intel 80386, which could operate with a clock speed up to 33 MHz. The bus was 32 bits wide, and the chip supported a 32-bit address field. This chip contained some 275,000 transistors, and was perhaps the transformative chip that shifted the personal computer from the periphery of the IT environment to the mainstream. This chip took on the mainframe computer and won. The evolving architecture of the late 1980s was shifting from a central processing center and a cluster of basic peripheral devices to one of a cluster of personal desktop computers.

The desktop personal computer environment enabled computing power to be treated as an abundant commodity, and with the desktop computer came numerous interface systems that allowed users to treat their computer screens in a manner that was analogous to a desktop. Information was organized in ways that had a visual counterpart, and applications interacted with the users in ways that were strongly visual. The approach pioneered by the Xerox Star workstation in the late 1970s and brought to the consumer market through the Apple Lisa and Macintosh systems were then carried across into the emerging "mainstream" of the desktop environment with Windows 2.0 in the late 1980s.

The state of the art of portability was still in the category of "luggable" rather than truly portable, and the best example of what was around at the time is the ill-fated Macintosh Portable, which like its counterpart in the portable phone space was the size of a briefcase and incredibly heavy.

Oddly enough, while the industry press was in raptures when it was released in 1989, it was a complete failure in the consumer market. The age of the laptop was yet to come.

One major by-product in this shift in the computing environment to a distributed architecture was a major shift in the attention to networking, and at the same time as there was a large-scale shift in the industry from mainframes to personal computers, there were also numerous major changes in the networked environment.

The Networking Environment of the Late 1980s

A networking engineer in the late 1980s was probably highly conversant in how to network serial terminals to mainframes. The pin-outs in the DB-25 plug used by the RS-232 protocol was probably one of the basic ABCs of computer networking. At that time much of the conventional networked environment was concerned with connecting these terminal devices to mainframes, statistical multiplexors, and terminal switches, and serial switch suppliers such as Gandalf and Micom were still important in many large-scale computing environments.

At the same time, another networking technology was emerging—initially fostered by the need to couple high-end workstations with mainframes—and that was *Ethernet*. Compared to the kilobits per second typically obtained by running serial line protocols over twisted pairs of copper wires, the 10-Mbps throughput of Ethernet was blisteringly fast. In addition, Ethernet could span environments with a diameter of around 1500 meters, and with a certain amount of tweaking or with the judicious use of Ethernet bridges and fibre-optic repeaters this distance could be stretched out to 10 km or more.

Ethernet heralded a major change in the networked environment. No longer were networks hub-and-spoke affairs with the mainframe system at the center. Ethernet supplied a common bus architecture that supported any-to-any communications. Ethernet was also an open standard, and many vendors were producing equipment with Ethernet interfaces. In theory, these interfaces all interoperated, at least at the level of passing Ethernet frames across the network (aside from a rather nasty incompatibility between this original Digital-Intel-Xerox specification and the IEEE 802.3 "standardized" specification!).

However, above the basic data framing protocol the networked environment was still somewhat chaotic. I recall the early versions of the multiprotocol routers produced by Proteon and Cisco supported more than 20 networking protocols! There was *DECnet*, a proprietary network protocol suite from the Digital Equipment Corporation, which at around 1987 had just released Phase IV, and was looking toward a Phase V release that was to interoperate with the International Organization for Standardization's *Open Systems Interconnection* (OSI) protocol suite[1] (more on this subject a bit later).

There was IBM's *Systems Network Architecture* (SNA), which was a hierarchical network that supported a generic architecture of remote job entry systems clustered around a central service mainframe. There was the *Xerox Network Services* (XNS) protocol used by Xerox workstations. Then there were Apollo's *Network Computing Architecture* (NCA) and Apple's *AppleTalk*. And also in this protocol mix was the *Transmission Control Protocol/Internet Protocol* (TCP/IP) protocol suite, used at that time predominately on UNIX systems, although implementations of TCP/IP for Digital's VAX/VMS system were very popular at the time. A campus Ethernet network of the late 1980s would probably see all of these protocols, and more, being used concurrently.

And there was the ISO-OSI protocol suite, which existed more as a future protocol suite than as a working reality at the time.

The ISO-OSI and TCP/IP protocol suites were somewhat different from the others that were around at the time because both were deliberate efforts to answer a growing need for a vendor-independent networking solution. At the time the IT environment was undergoing a transition from the monoculture of a single vendor's comprehensive IT environment—which bundled the hardware of the mainframe, network, peripherals, terminals, and the software of the operating system, applications, and network all into the one bundle—into a piecemeal environment that included a diverse collection of personal workstations, desktop computers, peripherals, and various larger minicomputers and mainframe computers in one environment. What was needed was a networking technology that was universally supported on all these various IT assets. What we had instead was a more piecemeal environment. Yes, it was possible to connect most of these systems into a common Ethernet substrate, but making A talk to B was still a challenge, and various forms of protocol translation units were also quite commonplace at the time. What the industry needed was a vendor-independent networking protocol, and there were two major contenders for this role.

ISO-OSI and TCP/IP

The ISO-OSI protocol suite was first aired in 1980. It was intended to be an all-embracing protocol suite that embraced both the IEEE 802.3 Ethernet protocols and the X.25 packet switching protocols that were favoured by many telephony operators as their preferred wide-area data services solution. The ISO-OSI network layer included many approaches, including the telephony sector's *Integrated Service Digital Network* (ISDN), a *Connection-Oriented Network Service* (CONS), a virtual circuit function based largely on X.75 that was essentially the "call-connection" function for X.25, and a *Connectionless Network Service* (CLNS), based loosely on the IP protocol with the use of the *End System-to-Intermediate System Routing Exchange Protocol* (ES-IS) routing protocol.

Above the network layer were numerous end-to-end transport protocols, notably *Transport Protocol Class 4* (TP4), a reliable connection-oriented transport service, and *Transport Protocol Class 0* (TP0), a connectionless packet datagram service. Above this layer was a Session Layer, X.215, used by the TP4 CONS services, and a Presentation Layer, defined using the *Abstract Syntax Notation One* (ASN.1) syntax.

ISO-OSI included numerous application-level services, including *Virtual Terminal Protocol* (VTP) for virtual terminal support, *File Transfer Access And Management* (FTAM) for file transfer, *Job Transfer And Management* (JTAM) for batch job submission, *Message Handling System* (MHS, also known as X.400) for electronic mail, and the X.500 Directory service. ISO-OSI also included a *Common Management Information Protocol* (CMIP). ISO-OSI attempted to be everything to everybody, as evidenced by the "kitchen sink" approach adopted by many of the OSI standardization committees at the time.

When confronted by many technology choices, the committees apparently avoided making a critical decision by incorporating both approaches into the standard. The most critical decision in this protocol suite was the inclusion of both connection-oriented and connectionless networking protocols. They also used session and presentation layer protocols, whose precise role was a mystery to many! ISO-OSI was a work-in-progress at the time, and the backing of the telephone sector, coupled with the support of numerous major IT vendors, gave this protocol an aura of inevitability within the industry. Whatever else was going to happen, there was the confident expectation that the 1990s would see all computer networks move inevitably to use the ISO-OSI protocol suite as a common, open, vendor-neutral network substrate.

If the ISO-OSI had a mantra of inevitably, the other open protocol suite of the day, the TCP/IP protocol suite, actively disclaimed any such future ambitions. TCP/IP was thought of at the time as an experiment in networking protocol design and architecture that ultimately would go the way of all other experiments, and be discarded in favor of a larger and more deliberately engineered approach. Compared to the ISO-OSI protocols, TCP/IP was extremely "minimalist" in its approach. Perhaps the most radical element in its design was to eschew the conventional approach at the time of building the network upon a reliable data link protocol. For example, in DECnet Phase IV, the data link protocol, *Digital Data Communications Message Protocol* (DDCMP), performed packet integrity checks and flow control at the data link level. TCP/IP gracefully avoided this problem by allowing packets to be silently dropped by intermediate data switches, or corrupted while in flight. It did not even stipulate that successive packets within the same end-to-end conversation follow identical paths through the network.

Thus the packet switching role was radically simplified because now the packet switch did not need to hold a copy of transmitted packets, nor did it need to operate a complex data link protocol to track packet transmission integrity and packet flow control. When a switch received a packet, it forwarded the packet based on a simple lookup of the destination address contained in the packet into a locally managed forwarding table. Or it discarded the packet.

The second radical simplification in TCP/IP was the use of real-time packet *fragmentation.* Previously, digital networks were constructed in a "vertically integrated" manner, where the properties of the lower layers were crafted to meet the intended application of the network. Little wonder that the telephone industry put its support behind X.25, which was a reliable unsynchronized digital stream protocol. If you wanted low levels of jitter, you used a network with smaller packet sizes, whereas higher packet sizes improved the carriage efficiency. Ethernet attempted to meet this wide variance in an agnostic fashion by allowing packets of between 64 and 1500 octets, but even so there were critics who said that for remote terminal access the smallest packets were too large, and for large-scale bulk data movement the largest packets were too small. *Fiber Distributed Data Interface* (FDDI), the 100-Mbps packet ring that was emerging at the time as the "next thing" as commodity high-speed networking used a maximum size of 4000 octets packets in an effort to improve carriage efficiency, whereas the *Asynchronous Transfer Mode* (ATM) committee tried to throw a single-packet-size dart at the design board and managed to get the rather odd value of 53 octets!

IP addressed this problem by trying to avoid it completely. Packets could be up to 64,000 octets long, and if a packet switch attempted to force a large packet through an interface that could not accept it, the switch was allowed to divide the packet into appropriately sized autonomous fragments. The fragments were not reassembled in real time: that was the role of the ultimate receiver of the packets.

As an exercise in protocol design, IP certainly showed the elegance of restraint. IP assumed so little in terms of the transmission properties of the underlying networks that every packet was indeed an adventure! But IP was not meant to be the protocol to support the prolific world of communicating silicon in the coming years. This protocol and the IP networks that were emerging in the late 1980s were intended to be experiments in networking. There was a common view that the lessons learned with experience of operating high-speed local networks and wide-area networks using the TCP/IP protocol suite would inform the larger industry efforts. The inclusion of IP-based technologies in the ISO-OSI protocol suite[4] was a visible instantiation of this proposed evolutionary approach.

While these two protocol suites vied with each other for industry attention at the time, there was one critical difference: It was a popular story at the time that the ISO-OSI protocol suite was a stack of paper some 6 feet high, which cost many hundreds of dollars to obtain, with no fully functional implementations, whereas the TCP/IP protocol suite was an open-sourced and openly available free software suite without any documentation at all. Many a jibe at the time characterized the ponderous approach of the ISO-OSI approach as "vapourware about paperware," while the IP effort, which was forming around the newly formed *Internet Engineering Task Force* (IETF), proclaimed itself to work on the principle of "rough consensus and running code."

Local- and Wide-Area Networking

The rise of Ethernet networks on campuses and in the corporate world in the late 1980s also brought into stark visibility the distinction between local- and wide-area networking.

In the local-area network, Ethernet created a new environment of "seamless connectivity." Any device on the network could provide services to any other device, and the common asset of a 10-Mbps network opened up a whole new set of computing possibilities. Data storage could be thought of as a networked resource, so desktop computers could access a common storage area and complement it with local storage, and do so in a way that the distinction between local resources and shared networkwide resources was generally invisible. The rich computing environment of visualizing the application, popularized by both the Macintosh and Windows 2.0, complemented a rich networked environment where rather than bringing a user into the location that had both the data and the computing resources, the model was invested, and the user was able to exclusively use the local environment and access the remote shared resources through networking capabilities integrated into the application environment. Local-area networking was now an abundant resource, and the industry wasted no time on exploiting this new-found capability.

But as soon as you wanted to venture further than your *Local-Area Network* (LAN), the picture changed dramatically. The wide-area networking world was provisioned on the margins of oversupply of the voice industry, and the services offered reflected the underlying substrate of a digital voice circuit. The basic unit of a voice circuit was a 64-kbps channel, which was "groomed" into a digital circuit of either 56 or 48 kbps, depending on the particular technology approach used by the voice carrier. Higher capacities (such as 256 or 512 kbps) were obtained by multiplexing individual circuits together. Even high-capacity circuits were obtained by using a voice trunk circuit, which was either 1.5 (T1) or 2.048 Mbps (E1), again depending on the digital technology used by the voice carrier. Whereas the LANs were now supporting an any-to-any mode of connection, these *Wide-Area Networks* (WANs) were constructed using point-to-point technologies that were either statically provisioned or implemented as a form of "on-demand" virtual circuit (X.25).

In the late 1980s users' patience was running thin over having to use an entirely different protocol suite for the wide area as distinct from the local area. Often the wide area required the use of different applications with different naming and addressing conventions. One approach used by many Ethernet switch vendors was to introduce the concept of an *Ethernet Serial Bridge*. This technology allowed a logical IEEE 802.3 Ethernet to encompass much larger geographic domains, but at the same time protocols that worked extremely efficiently in the local area encountered significant problems when passed through such supposedly "transparent" Ethernet serial bridges.

However, these bridge units allowed significantly larger and more complex networks to be built using Ethernet as the substrate. The Ethernet *Spanning Tree Algorithm* gained traction in order to allow arbitrary topologies of interconnected LANs to self-organize into coherent topologies that eliminated loops and allowed for failover resilience in the network.

What has changed, and what has stayed the same?

So what have we learned from this time?

In the intervening period ISO-OSI waned and eventually disappeared, without ever having enjoyed widespread deployment and use. Its legacy exists in numerous technologies, including the X.500 Directory Service, which is the substrate for today's *Lightweight Directory Access Protocol* (LDAP) Directory Services. Perhaps the most enduring legacy of the ISO-OSI work is the use of the "layered stack" conceptual model of network architectures. These days we refer to "Layer 2 Virtual LANs (VLANs)" and "Layer 3 Virtual Private Networks (VPNs)" perhaps without appreciating the innate reference to this layered stack model.

Of course the ISO-OSI protocol suite was not the only casualty of time. DECnet is now effectively an historic protocol, and Novell's *NetWare* has also shifted out of the mainstream of networking protocols. Perhaps it may be more instructive to look at those technologies that existed at the time that have persisted and flourished so that they now sit in the mainstream of today's networked world.

Ethernet has persisted, but today's Ethernet networks share little with the technology of the original IEEE 802.3 *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) 10-Mbps common bus network. The entire common bus architecture has been replaced by switched networks, and the notion of self-clocking packets was discarded when we moved into supporting Gbps Ethernets. What has persisted is the IEEE 802.3 packet frame format, and the persistence of the 1500-octet packet as the now universal lowest common factor for packet quantization on today's network. Why did Ethernet survive while other framing formats, such as *High-Level Data Link Control* (HDLC), did not?

I could suggest that it was a triumph of open standards, but HDLC was also an open standard. I would like to think that the use of a massive address space in the Ethernet frame, the 48-bit *Media Access Control* (MAC) address, and the use since its inception of a MAC address registry that attempted to ensure the uniqueness of each Ethernet device were the most critical elements of the longevity of Ethernet.

Indeed not only has UNIX persisted, it has proliferated to the extent that it is ubiquitous, because it now forms the foundation of the Apple and Android products. Of the plethora of operating systems that still existed in the late 1980s, it appears that all that have survived are UNIX and Windows, although it is unclear how much of Windows 2.0 still exists in today's Windows 7, if anything.

And perhaps surprisingly TCP/IP has persisted. For a protocol that was designed in the late 1970s, in a world where megabits per second was considered to be extremely high speed, and for a protocol that was ostensibly experimental, TCP/IP has proved to be extremely persistent. Why? One clue is in the restrained design of the protocol, where, as we have noted, TCP/IP did not attempt to solve every problem or attempt to be all things for all possible applications. I suspect that there are two other aspects of TCP/IP design that contributed to its longevity.

The first was a deliberate approach of modularity in design. TCP/IP deliberately pushed large modules of functions into distinct subsystems, which evolved along distinct paths. The routing protocols we use today have evolved along their own paths. Also the name space and the mapping system to support name resolution has evolved along its own path. Perhaps even more surprisingly, we have had the rate control algorithms used by TCP, the workhorse of the protocol suite, evolve along its own path.

The second aspect is use of what was at the time a massively sized 32-bit address space, and an associated address registry that allowed each network to use its own unique address space. Like the Ethernet 48-bit MAC address registry, the IP address registry was, in my view, a critical and unique aspect of the TCP/IP protocol suite.

Failures

What can we learn from the various failures and misadventures we have experienced along the way?

Asynchronous Transfer Mode (ATM) was a technology that despite considerable interest from the telephone operators proved to be too little too late, and was ultimately swept aside in the quest for ever larger and ever cheaper network transmission systems. ATM appeared to me to be perhaps the last significant effort to invest value into the network through allowing the network to adapt to the various differing characteristics of applications.

The underlying assumption behind this form of adaptive networking is that attached devices are simply incapable of understanding and adapting to the current state of the network, and it is up to the network to contain sufficient richness of capability to present consistent characteristics to each application. However, our experience has been quite the opposite, where the attached devices are increasingly capable of undertaking the entire role of service management, and complex adaptive networks are increasingly seen as at best meaningless duplication of functions, and at worst as an anomalous network behavior that the end device needs to work around. So ATM failed to resonate with the world of data networking, and as a technology it has waned. In the same way subsequent efforts to equip IP networks with *Quality of Service* (QoS) responses, or the much-hyped more recent *Next-Generation Networking* (NGN) networking efforts have been failures, for much the same basic reasons.

Fiber Distributed Data Interface (FDDI) also came and went. Rings are notoriously difficult to engineer, particularly in terms of managing a coherent clock across all attached devices that preserves the circumference of the ring, as measured in bits on the wire. From its earlier lower-speed antecedents in the 4-Mbps token, the 100-Mbps FDDI ring attracted considerable interest in the early 1990s. However, it was in effect a dead end in terms of longer-term evolution—the efforts to increase the clock speed required either the physical diameter of the ring to shrink to unusable small diameters or the clock signal to be locked at extraordinarily high levels of stability that made the cost of the network prohibitive. This industry appears to have a strong desire for absolute simplicity in its networks, and even rings have proved to be a case of making the networks too complex.

Interestingly, and despite all the evidence in their favor, the industry is still undecided about open technologies. TCP/IP, UNIX, and the Apache web platform are all in their own way significant and highly persuasive testaments to the power of open-source technologies in this industry, and a wide panoply of open technologies forms the entire foundation of today's networked environment. Yet, in spite of all this accumulated experience, we still see major efforts to promote closed, vendor-specific technologies into the marketplace. Skype is a case in point, and it is possible to see the iPhone and the Kindle in a similar light, where critical parts of the technology are deliberately obscured and aspects of the device behavior are deliberately sealed up or occluded from third-party interception.

The Next Twenty-Five Years

In wondering about the next 25 years, it may be interesting to look back ever further, to the early 1960s, and see what, if anything, has proved to be enduring from the perspective of the past 50 years. Interestingly, it appears that very little of that time, except for the annoying persistence of Fortran, and the ASCII keyboard as the ubiquitous input device, is still a part of today's networked environment. So over a 50-year time period much has changed in our environment.

But, interestingly, when we par down the period to the past 25 years, there is still much that has survived in the computing and networking environment. A Macintosh computer of the late 1980s looks eerily familiar, and although today's systems are faster, lighter, and a lot less clunky, there is actually very little that has changed in terms of the basic interface with the user. A Macintosh of that time could be connected to an Ethernet network, and it supported TCP/IP, and I suspect that if one were to resurrect a Mac system from 1988 loaded with *MacTCP* and connect it to the Internet today it would be frustratingly, achingly slow, but I would like to think that it would still work! And the applications that ran on that device have counterparts today that continue to use the same mechanisms of interaction with the user.

So if much of today's world was visible 25 years ago, then where are the aspects of change? Are we just touching up the fine-point details of a collection of very well established technologies? Or are there some basic and quite fundamental shifts underway in our environment?

It seems to me that the biggest change is typified in today's tablet and mobile phone computers, and in these devices it is evident that the metaphors of computing and interaction with applications are changing. The promise from 1968 in the film *2001: A Space Odyssey* of a computer that was able to converse with humans is now, finally, within reach of commodity computing and consumer products. But it is more than merely the novelty of a computer that can "talk." The constant search for computing devices that are smaller and more ubiquitous now means that the old paradigm of a computer as a "clever" but ultimately bulky typewriter is fading away. Today we are seeing modes of interaction that use gestures and voice, so that the form factor of a computer can become smaller while still supporting a functional and efficient form of interaction with the human user.

It is also evident that the pendulum of distribution and centralization of computing capability is swinging back, and the rise of the heavily hyped *Cloud*[5, 6] with its attendant collection of data centers and content distribution networks, and the simultaneous shrinking of the end device back to a "terminal" that allows the user to interact with views into a larger centrally managed data store held in this cloud, appears to be back in vogue once more.

It is an open question whether these aspects of today's environment will be a powerful and persistent theme for the next 25 years, or whether we will see other aspects of our environment seize industry momentum, so they are very much just a couple of personal guesses. Moore's Law has proved to be truly prodigious over the past 50 years. It has allowed us to pack what was a truly unbelievable computing capability and storage into astonishingly small packages and then launch them into the consumer market with pricing each year that appears to be consistently lower than the previous year.

If this property of packaging ever greater numbers of transistors into silicon chips continues for the next 25 years at the same rate, then it is likely that whatever happens in the next 25 years, the only limitation may well be our imagination rather than any intrinsic limitations of the technology itself.

For Further Reading

[0] The Charles Babbage Institute at the University of Minnesota has scanned the complete collection of *ConneXions—The Interoperability Report,* and it is available at this URL: **http://www.cbi.umn.edu/hostedpublications/Connexions/index.html**

[1] Starting in April 1989 (Volume 3, No. 4), *ConneXions* published a long-running series of articles under the general heading "Components of OSI," which described almost every aspect of this protocol suite. The same journal also published articles on many of the other technologies mentioned in this article, including FDDI, AppleTalk, and ATM.

[2] Vint Cerf, "A Decade of Internet Evolution," *The Internet Protocol Journal,* Volume 11, No. 2, June 2008.

[3] Geoff Huston, "A Decade in the Life of the Internet," *The Internet Protocol Journal,* Volume 11, No. 2, June 2008.

[4] International Organization for Standardization, "Final text of DIS 8473, Protocol for Providing the Connectionless-mode Network Service," RFC 994, March 1986.

[5] T. Sridhar, "Cloud Computing—A Primer Part 1: Models and Technologies," *The Internet Protocol Journal,* Volume 12, No. 3, September 2009.

[6] T. Sridhar, "Cloud Computing—A Primer Part 2: Infrastructure and Implementation Topics," *The Internet Protocol Journal,* Volume 12, No. 4, December 2009.

GEOFF HUSTON B.Sc., M.Sc., is the Chief Scientist at *Asia Pacific Network Information Centre* (APNIC), the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of numerous Internet-related books, was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001. E-mail: **gih@apnic.net**

Letter to the Editor

Dear Editor,

Who knew? Twenty-five years ago I started a tiny company that grew into Interop to spread the technical word about this funny thing we called *The Internet* and this really obscure thing called "TCP/IP." Back in the '70s, when the basic protocols were being created and experimented with, you were a high school kid in Norway and I was running a tiny group at SRI International and I let you use my machine across the ocean by using the ARPANET, the precursor to the Internet, because you seemed both smart and polite. Fifteen years later I decided to hire you to start a newsletter, *ConneXions—The Interoperability Report,* about the burgeoning Internet because of those properties and the perceived need to communicate monthly about the ins and outs of these simple but far-reaching technical protocols. You had the technical knowledge and good sense to enlist the brains of the real engineers in the field with real experience to further the knowledge of "all things Internet."

Who knew this would be still going on 25 years later? Your combination of passion and patience has produced an amazing record of ongoing expertise for the whole world to enjoy.

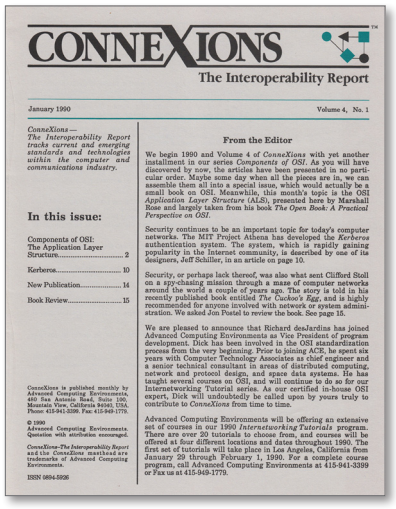
Thank you for being Ole!

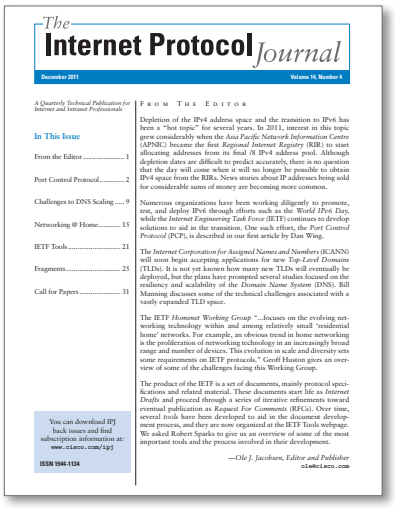
—Dan Lynch
Founder of Interop
dan@lynch.com

Thank you, Dan!

I appreciate your very kind words. I also want to take this opportunity to thank all of the contributors to this journal. We could not do this without you!

—Ole J. Jacobsen, Editor and Publisher
ole@cisco.com


CONNEXIONS
The Interoperability Report
January 1990 Volume 4, No. 1
In this issue:
From the Editor


The Internet Protocol Journal
December 2011 Volume 14, Number 4
From The Editor
In This Issue

Fragments

NIXI to Run New NIR in India

March saw the launch of a new *National Internet Registry* (NIR) for India, following the successful conclusion of talks between the *Asia Pacific Network Information Centre* (APNIC) and the Government of India.

The *Indian Registry For Internet Names And Numbers* (IRINN) will be run by the *National Internet Exchange of India* (NIXI) and serve ISPs within the country that wish to sign up. It is the result of a long collaboration between APNIC and NIXI, with APNIC staff sharing their expertise with NIXI, and NIXI officials putting together an impressive technical installation in preparation for the launch. The new registry was announced on the final day of APNIC 33, a technical conference conducted in conjunction with the *Asia Pacific Regional Internet Conference on Operational Technologies* (APRICOT 2012).

APNIC Executive Council Chairman, Akinori Maemura said of the announcement, "We are extremely happy that this process is heading towards a positive conclusion; which, on the other hand, is also a commencement of a new relationship. I would like to thank the NIXI team for their support and the hard work they have demonstrated in making this a reality."

Director General of APNIC, Paul Wilson commented, "We welcome the new National Internet Registry in India to the APNIC community. The Internet is a global community and IRINN, as the NIR is being called, should be part of that. I hope that many new Internet Services Providers will be formed in India, and they will always be able to choose between IRINN and APNIC for IP addresses. The market here is big enough and that kind of diversity will ensure better services and lower prices for all Indians."

APNIC has over 300 members locally, mostly Internet Services Providers and Telecommunication Communications companies, and over 6 million *Internet Protocol version 4* (IPv4) addresses were allocated in 2011. There are already 6 National Internet Registries in Asia in South Korea: (KISA KRNIC), Japan (JPNIC), China (CNNIC), Indonesia (IDNIC), Vietnam (VNNIC) and Taiwan (TWINIC). This is out of 56 economies in the Asia Pacific region.

"It's really about what is a better fit for the individual organizations. Typically we tend to see larger organizations prefer a regional service, especially those who operate in multiple economies to maintain an account with APNIC," said Paul Wilson.

NIXI is a not-for-profit organization, set up for peering of ISPs among themselves for the purpose of routing domestic traffic within India, instead of routing it through international peering points, thereby resulting in reduced latency and reduced bandwidth charges for ISPs. NIXI is managed and operated on a neutral basis, in line with the best practices for such initiatives globally.

Internet Hall of Fame Advisory Board Named

The *Internet Society* (ISOC) recently announced that in conjunction with its 20th anniversary celebration, it is establishing an annual *Internet Hall of Fame* program to honor leaders and luminaries who have made significant contributions to the development and advancement of the global Internet.

Inaugural inductees will be announced at an Awards Gala during the ISOC's *Global INET 2012* conference in Geneva, Switzerland, April 22–24, 2012, **www.internetsociety.org/globalinet**

"There are extraordinary people around the world who have helped to make the Internet a global platform for innovation and communication, spurring economic development and social progress," noted ISOC CEO Lynn St. Amour. "This program will honor individuals who have pushed the boundaries to bring the benefits of a global Internet to life and to make it an essential resource used by billions of people. We look forward to recognizing the achievements of these outstanding leaders."

ISOC has convened an Advisory Board to vote on the inductees for the 2012 Internet Hall of Fame inauguration. The Advisory Board is a highly-qualified, diverse, international committee that spans multiple industry segments and backgrounds. This year's Advisory Board members include:

- Dr. Lishan Adam, ICT Development Researcher, Ethiopia
- Chris Anderson, Editor-in-Chief, *WIRED Magazine*
- Alex Corenthin, Directeur des Systemes d'Information, University Cheikh Anta Diop of Dakar/Chair, Internet Society Senegal Chapter
- William Dutton, Professor of Internet Studies, Oxford Internet Institute
- Joichi Ito, Director, MIT Media Lab
- Mike Jensen, Independent ICT Consultant, South Africa
- Aleks Krotoski, Technology Academic/Journalist/Author
- Loic Le Meur, Founder & CEO, LeWeb
- Mark Mahaney, Internet Analyst, Citigroup
- Dr. Alejandro Pisanty, Professor at National University of Mexico/ Chair of Internet Society Mexico Chapter
- Lee Rainie, Director, Pew Research Center's Internet & American Life Project
- Jimmy Wales, Co-founder, Wikipedia

"We are extremely grateful to our distinguished Advisory Board members who have donated their time, energy, and expertise to this program," St. Amour added. "The breadth of their experiences and the diversity of their perspectives are invaluable, and we truly appreciate their participation."

The Internet Society is the trusted independent source for Internet information and thought leadership from around the world. With its principled vision and substantial technological foundation, the Internet Society promotes open dialogue on Internet policy, technology, and future development among users, companies, governments, and foundations. Working with its members and Chapters around the world, the Internet Society enables the continued evolution and growth of the Internet for everyone.

For more information, see: **http://www.internetsociety.org**

IETF Journal Now Available by Subscription

The *IETF Journal* provides anyone with an interest in Internet standards an overview of the topics being debated by the *Internet Engineering Task Force* (IETF), and also helps facilitate participation in IETF activities for newcomers.

The *IETF Journal* aims to provide an easily understandable overview of what is happening in the world of Internet standards, with a particular focus on the activities of the IETF Working Groups. Each issue highlights hot issues being discussed in IETF meetings and on the IETF mailing lists.

Visit *The IETF Journal* on the Web at **www.internetsociety.org/ietfjournal** to see the latest edition, or to subscribe to the e-mail edition or have it delivered as a hardcopy, visit:
http://www.internetsociety.org/ietfjournal-subscribe

The *IETF Journal* is an Internet Society publication produced in cooperation with the IETF.

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
PERMIT No. 5187
SAN JOSE, CA

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com