The Internet Protocol Journal

September 2003 Volume 6, Number 3

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From The Editor

The task of adding security to Internet protocols and applications is a large and complex one. From a user's point of view, the security-enhanced version of any given component should behave just like the old version, just be "better and more secure." In some cases this is simple. Many of us now use a *Secure Shell Protocol* (SSH) client in place of *Telnet,* and shop online using the secure version of HTTP. But there is still work to be done to ensure that *all* of our protocols and associated applications provide security. In this issue we will look at *routing,* specifically the *Border Gateway Protocol* (BGP) and efforts that are underway to provide security for this critical component of the Internet infrastructure. As is often the case with emerging Internet technologies, there exists more than one proposed solution for securing BGP. Two solutions, S-BGP and soBGP, are described by Steve Kent and Russ White, respectively.

The Internet gets attacked by various forms of viruses and worms with some regularity. Some of these attacks have been quite sophisticated and have caused a great deal of nuisance in recent months. The effects following the *Sobig.F* virus are still very much being felt as I write this. Tom Chen gives us an overview of the trends surrounding viruses and worms.

Closely related to the virus attacks is *spam.* Unfortunately, I know of no complete technical, or even legal, solutions to this growing problem, but I would love to hear your views and solutions. Send your comments to: **ipj@cisco.com**, but don't use the string "spam" in the subject field or it may get filtered out!

Following Geoff Huston's opinion piece "The Myth of IPv6" in our previous issue, we received a response from *The IPv6 Forum.* The article is entitled "IPv6 Behind the Wall" and is by Jim Bound.

I was very pleased to hear that professor Peter T. Kirstein of University College London had been awarded the Internet Society's *Jonathan B. Postel Service Award* for 2003. I have known Peter since about 1977, when we collaborated on SATNET packet voice conferences between Oslo, London, Boston, and Marina del Rey. Peter is truly an Internet pioneer. (See "Fragments," page 41).

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

Securing the Border Gateway Protocol

by Stephen T. Kent, BBN Technologies

Routing in the public Internet is based on a distributed system composed of many routers, grouped into management domains called *Autonomous Systems* (ASes). ASes are operated by *Internet Service Providers* (ISPs) and by multihomed subscribers. (Throughout the remainder of this article, for brevity, we will talk in terms of ISPs, usually omitting references to multihomed subscribers.) Routing information is exchanged between ASes using the *Border Gateway Protocol* (BGP)[1], via UPDATE messages.

BGP is used in two different contexts. *External BGP* (eBGP) propagates routes between ISPs. BGP also is used within an AS to propagate routes acquired from other ASes. This latter use is referred to as *internal BGP* (iBGP). eBGP is the primary focus of this article, because failures of eBGP can adversely affect large portions of the Internet, well beyond the administrative boundary of the source of the failure. Nonetheless, some ISPs have expressed interest in protecting the distribution of routes within an ISP. The security technology discussed in this article can be used to secure iBGP, but eBGP is the focus of this article. We use the term "BGP" to refer to eBGP throughout the article.

BGP is highly vulnerable to a variety of attacks[2]. In some cases, this vulnerability arises because of a lack of integrity and authentication for BGP messages. However, the more substantive and harder problem is the lack of a secure means of verifying that BGP traffic is authorized, a concept explored in more detail in this article. In April 1997, BBN began work on the security architecture described here, a system we refer to as *S-BGP,* to address the vulnerabilities of BGP. This article begins by reviewing the problem, discusses a model for correct operation of BGP, presents a threat model, and states the goals and assumptions that underlie our proposed security architecture.

Before we begin the discussion of BGP in more detail, a few definitions are in order. A *route* is defined as an *address prefix* and a set of *path attributes*. One of the path attributes is an AS path, and that is the primary focus of BGP security considerations. The AS path specifies the sequence of ASes that subscriber traffic should traverse if forwarded via this route. When propagating an UPDATE to a neighboring AS, the BGP router prepends its AS number to the sequence, and may update certain other path attributes. The first AS included in the path is referred to as the *origin AS*.

Each BGP router (other than at the edges of the Internet) maintains a complete routing table, capable of routing traffic to any reachable destination, and sends its best route for each prefix to each neighbor. In BGP, "best" is very locally defined. The BGP route selection algorithm has few criteria that are universal, thus limiting the extent to which any security mechanism can detect and reject "bad" routes emitted by a neighbor.

Each ISP makes use of local policies that it need not disclose, and this gives BGP route selection a "black box" flavor, which has significant adverse implications for security.

Correct Operation of BGP

Security for BGP should be defined as the correct operation of BGP routers. This definition is based on the observation that any successful attack against BGP will result in other than correct operation, presumably yielding degraded routing. Correct operation of BGP depends upon the integrity, authenticity, and timeliness of the routing information it distributes, as well as each BGP router processing, storing, and distributing this information in accordance with both the BGP specification and local routing policies. Many statements could be made in an effort to characterize correct operation, but they rest on two simple assumptions.

First, control (vs. subscriber traffic) communication between neighbor BGP routers must be authenticity and integrity secure. This is easily achieved through the use of a point-to-point security protocol capable of protecting BGP traffic; for example, *IP Security* (IPSec). Second, BGP routers must execute the route selection algorithm correctly and communicate the results. There are two parts to this assumption: processing received UPDATEs, and generation and transmission of UPDATEs. In terms of an AS trying to protect itself against external attacks, correct operation of its own BGP routers is mostly a local security issue, but not an Internet-wide security issue. However, an AS should not rely on other ASes to operate properly; such reliance permits a failure in one AS to propagate to others, a domino failure effect. Thus it is important for a BGP router to be able to verify that each UPDATE it receives from a peer is valid (authorized) and timely.

The validity of an UPDATE message is based on four primary criteria:

- The router that sent the UPDATE was authorized to act on behalf of the AS it claims to represent; that is, the AS at the front of the AS path.
- The AS from which the UPDATE emanates was authorized by the preceding AS in the AS path (in the UPDATE message) to advertise the prefixes in the UPDATE.
- The first AS in the AS path was authorized, by the owner of the set of prefixes that are represented in the UPDATE, to advertise those prefixes.
- If the UPDATE withdraws one or more routes (specified by the prefixes for the routes), then the sender must have advertised each route prior to withdrawing it.

There are some limitations to the ability of any practical security mechanism to detect all BGP security failures. The local policy feature of BGP allows each ISP considerable latitude in how UPDATEs are processed, making it difficult for an external observer—for example, a router in a neighboring AS—to determine if a router is operating properly.

This is because such behavior might be attributed to local policies not visible outside an AS. To address such attacks, the semantics of BGP itself would have to change. Moreover, because UPDATEs do not carry sequence numbers, a BGP router can emit an UPDATE based on authentic, but old, information; for example, withdrawing or reasserting a route based on outdated information. Thus the temporal accuracy of UPDATEs, in the face of Byzantine failures, is hard to enforce, except in a very coarse fashion. (Simply speaking, a *Byzantine failure* is one in which a nominally trusted or authorized entity misbehaves.)

Threat Model and BGP Vulnerabilities

Routers exhibit both architectural and implementation vulnerabilities. Implementation vulnerabilities are the result of errors that arise in developing design details or coding; for example, translating the BGP specs into software. Architectural vulnerabilities permit various forms of attack, independent of implementation details, and thus are potentially more damaging, because they persist across all implementations. To make Internet routing robust, both forms of vulnerabilities must be addressed. BGP vulnerabilities can be exploited to cause improper routing or nondelivery of subscriber traffic, network congestion, and traffic delays. Misrouting attacks can be used to facilitate both passive and active wiretapping of subscriber traffic. Often an attack against BGP may be part of a larger attack against subscriber computers. For example, there have been BGP attacks that seek to misroute queries to *Domain Name System* (DNS) root servers, as part of an attack against subscriber systems.

BGP can be attacked in many ways. Communication between BGP peers can be subjected to active or passive wiretapping. The BGP software, configuration information, or routing databases of a router may be modified or replaced via unauthorized access to a router, or to a server or management workstation from which router software is downloaded. These latter attacks transform routers into hostile insiders, so security measures must address such Byzantine failures.

Improved physical and procedural security for network management facilities, and routers, and cryptographic security for BGP traffic between routers would help reduce some of these vulnerabilities. However, physical and procedural security is expensive and imperfect, and these countermeasures would not protect the Internet against accidental or malicious misconfiguration by operators, nor against attacks that mimic such errors. Misconfiguration of this sort has been a source of Internet outages in the past and seems likely to persist. Any security approach that relies on ISPs to act properly violates the "principle of least privilege" and leaves the Internet routing system vulnerable at its weakest link. In contrast, the security approach described in this article satisfies this principle, so that any attack on any component of the routing system is limited in its impact on the Internet as a whole.

Routers also are susceptible to resource exhaustion attacks based on delivery of large quantities of management traffic, BGP or otherwise. This vulnerability arises because these devices are designed with the not unreasonable model that management traffic is a very tiny percentage of all the traffic that arrives at a router. Router interfaces can deliver traffic to the management processor at very high rates, because they are designed to accommodate subscriber traffic flows. Solutions to this problem need to be generic, to accommodate all types of router management traffic, and thus are outside the scope of the BGP security measures discussed in this article.

Goals, Constraints, and Assumptions

Any proposed security architecture must exhibit dynamics consistent with the existing BGP system; for example, responding automatically to topology changes, including the addition of new networks, routers, and ASes. These actions take place on different time scales and have different scopes. For example, in the current BGP system, if an ISP replaces a failed router, the action can take place fairly quickly and has only local impact, because ISPs are not aware of the identity of routers in other, non-neighboring, ISPs. The issuance of new AS numbers, representing new nets, is not a fast process, nor is the allocation of new blocks of address space (new prefixes). But both of these actions are globally visible. Changes in routes also may have global impact, and they may occur very quickly.

Solutions also must scale in a manner consistent with the growth of the Internet. The countermeasures must be consistent with the BGP protocol standards and with the likely evolution of these standards. This includes packet size limits and features such as path aggregation, communities, and multiprotocol support (for example, *Multiprotocol Label Switching* [MPLS]). The security measures must be incrementally deployable; there cannot be a "flag day" when all BGP routers suddenly begin executing a new security protocol. It is desirable to not create new organizational entities that must be accepted as authorities by ISPs and subscribers, in order to make routing secure.

S-BGP Architecture

S-BGP consists of four major elements:

- A *Public Key Infrastructure* (PKI) that represents the ownership and delegation of address prefixes and AS numbers
- *Address Attestations* that the owner of a prefix uses to authorize an AS to originate routes to the prefix
- *Route Attestations* that an AS creates to authorize a neighbor to advertise prefixes
- *IPSec* for point-to-point security of BGP traffic transmitted between routers

These elements are used by an S-BGP router to secure communication with neighbors, and to generate and validate UPDATE messages relative to the authorization model represented by the PKI and address attestations. Together, the combination of these security mechanisms prevents a compromised AS from propagating erroneous routing data to other, secured ASes. Each element is described in more detail in the following section.

S-BGP Public Key Infrastructure

S-BGP uses a PKI based on X.509 (v3) certificates to enable routers to validate the authorization of other routers to represent ASes (ISPs). The PKI also allows routers to verify the authorization of each ISP as the owner of one or more prefixes (contiguous blocks of address space). This PKI was described in[14], and the reader is referred to that paper for additional details. The PKI parallels the existing IP address and AS number assignment delegation system and takes advantage of this infrastructure. Because the PKI mirrors existing infrastructure, it avoids most of the "trust" issues that often complicate the creation of a PKI. This PKI is unusual in that it emphasizes authorization, not authentication. The names used in the certificates in this PKI are not employed to determine whether a given ISP or router is authorized to do anything, and the names are not even meaningful outside of S-BGP.

S-BGP calls for a certificate to be issued to each ISP (or subscriber) that owns (more properly, has a right to use) a portion of the IP address space. This certificate is issued through the same procedures employed for address allocation, starting with the *Internet Assigned Numbers Authority* (IANA) and continuing through a *Regional Internet Registry* (RIR), and, if applicable, an ISP. If an ISP owns multiple prefixes, we issue a single certificate containing a list of prefixes, to minimize the number of certificates in the system. The PKI represents address-space ownership by binding prefixes to a public key belonging to the ISP to which the prefixes have been assigned. Each certificate contains a private extension that specifies the set of prefixes that has been allocated to the ISP. Certificates issued under this PKI also represent the binding between an ISP and the AS numbers allocated to it. The PKI allows each ISP to issue certificates to its routers, certifying that these routers represent the ISP and hence, the ASes owned by the ISP. Here too, the PKI parallels the existing AS allocation system; that is, the IANA allocates AS numbers to RIRs, which in turn assign AS numbers to ISPs that run S-BGP.

Attestations

An *attestation* is a digitally signed datum asserting that its target (an AS) is authorized by the signer (an ISP) to advertise a path to one or more specified prefixes. There are two types of attestations, address and route, which share a common format. For an *Address Attestation* (AA), the signer is the ISP or subscriber that controls the prefixes in the AA, and the target is a set of ASes that the ISP/subscriber authorizes to originate a route to the prefixes. AAs are relatively static data items, because relationships between address-space owners and ISPs change relatively slowly.

For a *Route Attestation* (RA), the signer is an S-BGP router (operating on behalf of an ISP), and the target is an AS or set of ASes, representing the neighbors to which the UPDATE containing the RA will be sent. RAs, unlike AAs, are very dynamic, possibly changing for each transmitted UPDATE.

UPDATE Validation

Attestations and certificates are used by S-BGP routers to validate routes asserted in UPDATE messages; that is, to verify that the first AS in the route has been authorized to advertise the prefixes by the prefix owner(s), and that each subsequent AS has been authorized to advertise the route for the prefixes by the preceding AS in the route. To validate a route received from AS_n, AS_{n+1} requires:

- An AA for each organization owning a prefix represented in the UPDATE (not for prefixes in the UPDATE that represent routes being withdrawn)
- A certified public key for each organization owning a prefix in the UPDATE
- An RA corresponding to each AS along the path (AS_n to AS_1), where the RA generated and signed by the router in AS_n encompasses the *Network Layer Reachability Information* (NLRI) and the path from AS_{n+1} through AS_1
- A certified public key for each S-BGP router that signed an RA along the path (AS_n to AS_1), to check the signatures on the corresponding RAs

An S-BGP router verifies that the advertised prefixes and the origin AS are consistent with AA information. The router verifies the signature on each RA and verifies the correspondence between the signer of the RA and the authorization to represent the AS in question. There also must be a correspondence between each AS in the path and an appropriate RA. If all of these checks pass, the UPDATE is valid.

AAs are not used to check withdrawn routes in an UPDATE. Use of IPSec to secure communication between each pair of S-BGP routers, plus the fact that BGP uses a separate *Adjacency Routing Information Base* (Adj-RIB-In) for each neighbor, ensures that only the advertiser of a route can withdraw it.

Distribution of S-BGP Data

Each S-BGP router must have the public keys required to validate the RAs in UPDATEs, a scenario that translates into securely distributed keys for every router that implements S-BGP (and that is reachable via an S-BGP path). Each router also needs access to all AA information, to verify that the origin AS is authorized to originate a route to the prefixes in the UPDATE. S-BGP does not distribute certificates, *Certificate Revocation Lists* (CRLs), or AAs via UPDATE messages; transmission of these items via UPDATEs would be very wasteful of bandwidth, because each BGP router would receive many redundant copies from its neighbors.

Also, an UPDATE is limited to 4096 bytes and thus generally could not carry all of this data for the route represented by the UPDATE. Instead, S-BGP distributes this data to routers via out-of-band means. The data is relatively static and thus is a good candidate for caching and incremental update. Moreover, the certificates and AAs can be validated and reduced to a more compact format by ISP operation centers prior to distribution to routers. This avoids the need for each router to perform this processing, saving both bandwidth and storage space. It also means that routers do not need to be able to parse X.509 certificates and validate certificate paths for S-BGP purposes, although some capability in this area may be required for IPSec key management.

S-BGP uses *repositories* for distribution of this data. We initially described a model in which a few replicated, loosely synchronized repositories were operated by the RIRs. Discussions with ISPs suggest a model in which major ISPs and Internet exchanges operate repositories, and smaller ISPs and subscribers make use of these repositories. In either model, each ISP periodically, for example daily, uploads new/changed certificates, its current CRL, and AAs. Each ISP also downloads all of this data for all other ISPs that are running S-BGP. The repositories periodically transfer new data to one another to maintain loose synchronization. ISPs process the repository information to create more compact files that contain the AA data and the public keys and prefix and AS data from the certificates, but none of the certificate management information or CRLs. These resulting "extracted" files are transferred to the routers executing S-BGP under the control of the ISP.

Because certificates, AAs, and CRLs are signed and carry validity interval information, they require minimal additional security while in transit to or from a repository or while stored on a repository. Nonetheless, S-BGP employs the *Secure Sockets Layer* (SSL) protocol, with both client and server certificates, to protect access to the repositories, as a countermeasure to denial-of-service attacks. The simple, hierarchic structure of the PKI allows repositories to automatically effect access control checks on the uploaded data, for example, to prevent one ISP from accidentally or maliciously overwriting the certificates, CRLs, and AAs from another ISP.

Distribution of Route Attestations

S-BGP distributes RAs with BGP UPDATEs in a newly defined, optional, *transitive path attribute*. Because routes may change quickly, it is important that RAs accompany the UPDATEs that are validated using them. If any other means of distribution is employed for this data, there is a likelihood that the UPDATEs and the data will be out of synch, creating a conundrum for a router; that is, what should the router do when the UPDATE and the security data differ? RAs employ a compact encoding scheme to help ensure that they fit within the BGP packet size limits, even when route or address aggregation occurs. (S-BGP accommodates aggregation by explicitly including signed attribute data that otherwise would be lost when aggregation occurs.) An S-BGP router receiving an UPDATE from a peer caches the RAs with the route in the Adj-RIB for the peer, and in the *Local Routing Information Base* [Loc-RIB] (if the route is selected).

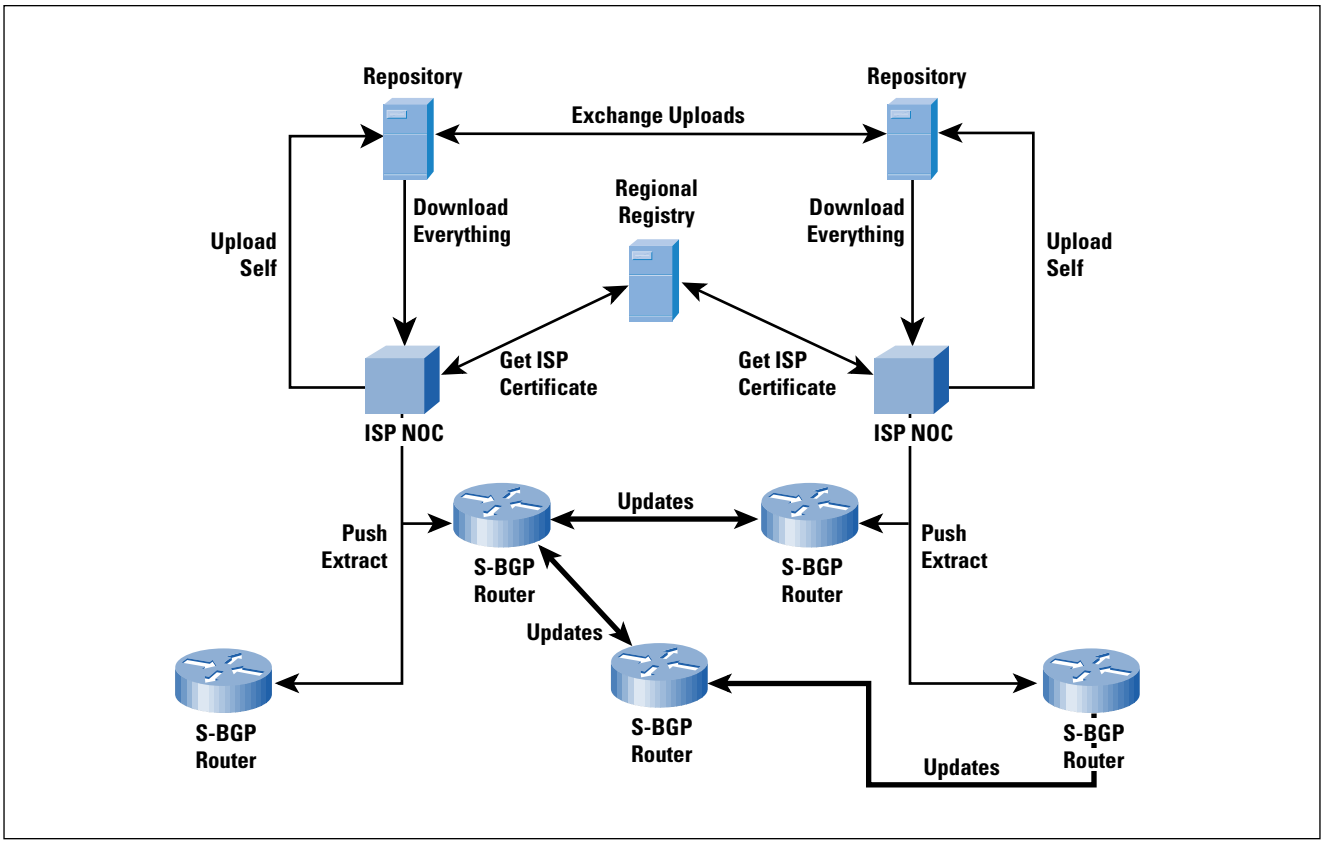
As noted in the following discussion, the bandwidth required to support in-band distribution of route attestations is negligible (compared to subscriber traffic).

Although the RA mechanism was designed to protect AS path data, it can also accommodate other new path attributes; for example, communities[11] and confederations[12]. Specifically, there is a provision to indicate what data, in addition to the AS path, is covered by the digital signature that is part of the RA.

Putting It All Together

Figure 1 illustrates how the major elements of S-BGP interact, using a simplified example. The figure shows two ISPs, each with a *Network Operations Center* (NOC), a repository, and three routers. A third ISP is represented by a single (S-BGP-enabled) router. Each ISP interacts with an RIR to acquire a certificate representing the prefixes and AS numbers assigned to the ISP. Each NOC interacts with a repository to upload data (certificates, CRLs, and AAs) from that ISP, and to download the same data acquired from all other ISPs. The repositories interact with one another to exchange uploaded ISP data, to make that data available to all other ISPs. Within an ISP, the NOC pushes a copy of the extracted certificate and AA data, produced from the downloads acquired from a repository, to each router. Routers exchange UPDATE messages, containing RAs, that enable validation of each received UPDATE.

Figure 1: S-BGP Element Interactions



IPSec and Router Authentication

S-BGP uses IPSec[6,7,8], specifically the *Encapsulating Security Payload* (ESP) protocol, to provide authentication, data integrity, and antireplay for all BGP traffic between neighboring routers. The *Internet Key Exchange* (IKE) protocol[9,10] is used for key management services in support of ESP. The S-BGP PKI includes certificates for IKE, separate from those used for RA processing.

The use of IPSec is preferable to the current option of the *Message Digest Algorithm 5* (MD5) TCP checksum option[15], in several respects. IPSec uses keyed hash functions in a way that is cryptographically more secure that the MD5 checksum option, and IKE provides automated key management, a feature sorely lacking in the option. Protecting BGP traffic at the IP layer, vs. the TCP layer, counters more vulnerabilities, because the TCP implementation is protected as well, for example, including SYN flooding and spoofed RSTs (resets), are rejected.

Residual Vulnerabilities in S-BGP

Despite the extensive security offered by S-BGP, architectural vulnerabilities exist that are not eliminated by its use. For example, an S-BGP router may reassert a route that was withdrawn earlier, even if the route has not been readvertised. The router also may suppress UPDATEs, including ones that withdraw routes. These vulnerabilities exist because BGP UPDATEs do not carry sequence numbers or time stamps that could be used to determine their timeliness. However, RAs do carry an expiration date and time, so there is a limit on how long an attestation can be misused this way. S-BGP restricts malicious behavior to the set of actions for which a router or AS is authorized, based on externally verifiable, authoritative constraints.

Performance and Operational Issues

In developing the S-BGP architecture, we paid close attention to the performance and operational impact of the proposed countermeasures, and reported our analysis in earlier papers. In preparing this article, we updated our data, utilizing a variety of sources; for example, the *Route Views* project. Although much data about BGP and associated infrastructure is available, other data is difficult to acquire in a fashion that is representative of a "typical" BGP router. This is because each AS in the Internet embodies a slightly different view of connectivity, as a result of local policy filters applied by other ASes.

It is important that the transmission, storage, and processing requirements imposed by S-BGP not be so great as to overwhelm routers. Each of these requirements must be analyzed separately.

The transmission of RAs in UPDATEs does significantly increase the size of these messages, by about 800 percent. However, because the volume of this traffic is minuscule relative to subscriber traffic, the increase is negligent. The set of files containing certificates, AAs, and CRLs would be about 75–85 MB. Daily transmission of these files between ISPs and repositories would not represent a significant increase in traffic volume for the Internet.

Although the transmission overhead is not a concern, storage of the RAs in each Adj-RIB and the Loc-RIB is a problem. The additional space required to hold these RAs is estimated at about 30–35 MB per peer, if S-BGP were fully deployed today. This is a modest amount of memory for a typical router with a few peers, but a significant amount of storage for routers at Internet exchanges, where a router may have tens or even hundreds of peers.

Thus the management CPU in a router might need a gigabyte or more of RAM under these conditions. (When a large ISP peers with many other ISPs at an exchange, the peering is not symmetric; that is, the large ISP accepts only a few routes from each of the smaller ISPs, filtering out the rest. Thus the amount of additional memory required for RAs in Adj-RIBs for each of these small ISP peers may be considerably less than for symmetric peer relationships.) This requisite memory seems modest by current workstation standards, but most deployed routers cannot be configured with this much memory.

The computational burden of router processing of RAs in UPDATEs is a function of the path length in each UPDATE and the rate at which UPDATEs arrive. The arrival rate is a function of the number of S-BGP peers the router sees, and the rate at which each peer sends UPDATEs. Our analysis suggests that the long-term (24-hour) UPDATE rate for a router with 30 peers is about 0.5 UPDATEs per second. On average, each UPDATE would contain about 3.7 RAs. We originally estimated the busy minute rate as about 10 times the average rate. At this rate, a router could probably perform the requisite signature verification in software (about 18 signature verifications per second). Recent evidence suggests a factor of 100–200 might be a better estimate, in light of experience with major worm attacks, and at that rate it would be hard for software to keep pace.

Heuristics are available to reduce this burden. Analysis shows that about 50 percent of all UPDATEs are sent as a result of route "flaps"; that is, transient communication failures that, when remedied, result in a return to the former route. Thus if a router maintained a depth-two cache for each Adj-RIB-In, it could avoid signature validation about 50 percent of the time. However, this would double the storage requirements for these RIBs, and that would exacerbate the storage problem cited previously.

Our previous analysis also assumed that receipt of each UPDATE would result in transmission of an UPDATE with one new signature. This was an oversimplification; a router generates and transmits an UPDATE only if the newly received route is "better" than the current best route (for the prefix), or if the best route for the prefix is withdrawn by the UPDATE. When a router has many peers, most of the UPDATEs it receives may not yield a better route, and thus will not trigger transmission of a new UPDATE.

On the other hand, when a router does select a new route, an UPDATE may be constructed and sent to each neighbor, requiring one signature per neighbor. This is because an RA specifies the AS number of the neighbor to which it is directed. It is possible to construct an RA that identifies the next hop as a set of AS numbers, corresponding to all the neighbors to which an UPDATE is authorized to be sent. The downside of this strategy is that it makes the RAs larger, contributing to the storage problem noted previously.

The observation made previously suggests a heuristic for UPDATE processing to mitigate signature validation costs. A router can defer validation of the RAs in any UPDATE that it receives, if the UPDATE would not represent a new best route. This optimization could be especially helpful for routers that receive the greatest number of UPDATEs; that is, routers with many neighbors. One might worry that this strategy allows an attacker to force processing, by sending what would be considered "very good" routes, but an S-BGP router could detect such fraudulent UPDATEs and could choose to drop its connection to a peer that behaved this way, in order to counter such an attack.

Initialization/reboot of a BGP router also results in a surge in UPDATE processing, and the deferred processing heuristic is applicable here too, even though reboots are relatively infrequent. Saving RIBs in nonvolatile storage addresses this problem. Most deployed routers do not have sufficient nonvolatile storage to adopt this strategy, but some do have hard drives that would easily accommodate the RIBs.

It is reasonable to assume that next-generation routers could be configured with enough RAM for the RIBs, but this analysis shows that full deployment is not feasible with the currently deployed router base. To add RAM, and possibly to add nonvolatile storage, router vendors will have to upgrade the processor boards where net management processing takes place. That suggests that addition of a crypto accelerator chip would be prudent as part of the board redesign process, for example, to deal with surge conditions noted previously.

Deployment and Transition Issues

Adoption of S-BGP requires cooperation among several groups. ISPs and subscribers running BGP must cooperate to generate and distribute AAs. Major ISPs must implement the S-BGP security mechanisms in order to offer significant benefit to the Internet community. The IANA and RIRs must enhance operational procedures to support generation of prefix and AS number allocation certificates. Router vendors need to offer additional storage in next-generation products, or offer ancillary devices for use with existing router products, and revise BGP software to support S-BGP.

There is some good news; S-BGP can be deployed incrementally. Only neighboring ASes receive full benefit from such deployment. Although we chose a transitive path attribute syntax to carry RAs, and thus it might be possible for non-neighbor ASes to exchange RAs, it seems likely that intervening ASes would not have sufficient storage for the RAs in their RIBs.

Also, the controls needed in routers to take advantage of noncontiguous deployment of S-BGP are quite complex, hence our suggestion that only contiguous deployment of S-BGP be attempted.

External routes received from S-BGP peers need to be redistributed within the AS, both to interior routers and to other border routers, in order to maintain a consistent and stable view of the exterior routes across the AS. Thus an AS must switch to using S-BGP for all its border routers at once, to avoid route loops within the AS.

Status

As of early 2003, an implementation of S-BGP has been developed and demonstrated on small numbers of workstations representing small numbers of ASes. We also developed software for a simple repository, and for NOC tools that support secure upload and download of certificates, CRLs, and AAs to and from repositories, and for certificate management for NOC personnel and routers. This suite of software, plus CA software from another *Defense Advanced Research Projects Agency* (DARPA) program, provide all of the elements needed to represent a full S-BGP system. All of this software is available in open source form.

Summary

S-BGP represents a comprehensive approach to addressing a wide range of security concerns associated with BGP. It detects and rejects unauthorized UPDATE messages, irrespective of the means by which they arise; for example, misconfiguration, active wiretapping, compromise of routers or management systems, etc. S-BGP is not perfect; it has a few residual vulnerabilities, but these pale in comparison to the security features S-BGP provides, and removal of these vulnerabilities would require more fundamental changes to BGP semantics.

The S-BGP design is based on a top-down security analysis, starting with the semantics of BGP and factoring in the wide range of attacks that have or could be launched against the existing infrastructure.

Acknowledgements

Many individuals contributed to the design and development of S-BGP, including Christine Jones, Charlie Lynn, Joanne Mikkelson, and Karen Seo.

References

[1] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995.

[2] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications,* Vol. 18, No. 4, April 2000.

[3] C. Villamizar, R. Chandra, and R. Govindan, "BGP Route Flap Damping," RFC 2439, November 1998.

[4] B.R. Smith, and J.J. Garcia-Luna-Aceves, "Securing the Border Gateway Routing Protocol," Proceedings of Global Internet '96, November 1996.

[5] S. Murphy, panel presentation on "Security Architecture for the Internet Infrastructure," Symposium on Network and Distributed System Security, April 1995.

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

[7] R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use with IPsec," RFC 2410, November 1998.

[8] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406, November 1998.

[9] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, November 1998.

[10] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, November 1998.

[11] R. Chandra, P. Traina, and T. Li, "BGP Communities Attribute," RFC 1997, August 1996.

[12] P. Traina, "Autonomous System Confederations for BGP," RFC 1965, June 1996.

[13] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol Extensions for BGP-4," RFC 2283, February 1998.

[14] K. Seo, C. Lynn, and S. Kent, "Public-Key Infrastructure for the Secure Border Gateway Protocol (S-BGP)," DARPA Information Survivability Conference and Exposition, June 2001.

[15] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option," RFC 2385, August 1998.

STEPHEN KENT received the S.M., E.E., and Ph.D. degrees in computer science from MIT, and a B.S. in mathematics from Loyola University of New Orleans. He has worked at BBN for over 25 years, where he serves today as Chief Scientist–Information Security. He served on the IAB for over a decade, and chaired the Privacy & Security Research Group of the IRTF and the PEM WG in the IETF, where he currently co-chairs the PKIX WG. He has served on several committees for the National Research Council, and chairs a committee on authentication and privacy for the NRC. His current work focuses on PKI issues, BGP security, and very high speed IP encryption. He is a Fellow of the ACM, and a member of the Internet Society and Sigma Xi. His e-mail address is: **kent@bbn.com**

Securing BGP Through Secure Origin BGP

by Russ White, Cisco Systems

Networks have come under increasing scrutiny in the area of security. Routing, the part of the network that provides information on how to reach destinations within the network, has been gaining attention from a security perspective as well. *The Internet Engineering Task Force* (IETF) has, in fact, formed a new working group, the *Routing Protocols Security Requirements Working Group* (**`http://www.rpsec.org`**), to analyze security in routing systems.

Of course, the biggest network in existence is the Internet, and the routing protocol that provides reachability and path information for the Internet is the *Border Gateway Protocol* (BGP), specified in RFC 1771. Several methods of securing the information carried within BGP have been proposed:

- *Internet Route Verification* (IRV), described in "Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing," Symposium on Network and Distributed Systems Security, February 2003, by Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick McDaniel, and Aviel Rubin. IRV relies on out-of-band communication with a route originator to verify the correctness of a route.
- S-BGP, described in the companion article and at: **`www.net-tech.bbn.com/projects/s-bgp`**
- *Domain Name System* (DNS)-based *Network Layer Reachability Information* (NLRI) origin *Autonomous System* (AS) verification in BGP, which is the oldest attempt at validating the information carried within BGP, is described in **`draft-bates-bgp4-nlri-orig-verif-00.html`**,

This article discusses *Secure Origin BGP* (soBGP), a solution recently proposed by a group (including me) mostly within Cisco Systems. We believe soBGP to be a deployable mechanism for validating the correctness and authorization of the data carried within BGP, and also for preventing the sorts of attacks resulting from misconfiguration or intentional insertion of bad data into the Internet routing system.

We address four goals when we consider security in terms of BGP:

- Is the AS originating the destination (prefix) authorized to advertise it? In other words, if a router receives an advertisement for the 10.1.1.0/24 network originating in AS65500, is there any way to verify that AS65500 is supposed to be advertising 10.1.1.0/24?
- Does the AS advertising the destination actually have a path to the destination? In other words, if a router is receiving an advertisement from a BGP peer in AS65501 that it can reach 10.1.1.0/24, is there any way to verify that AS65501 actually has a path to the AS origination 10.1.1.0/24?

- Is the peer advertising the route authorized by the originator, or owner, of the destination, to advertise a path to the destination?
- Does the path advertised by a peer AS fall within the policies the local network administrators have set forward? The most obvious issue is whether or not the AS Path advertised by the peer is an acceptable path to send the traffic along.

We argue elsewhere that the second two goals cannot be fully met within an operational internetwork, for many reasons; see **`draft-white-pathconsiderations-00.txt`** for further discussion on this point. In this article, then, we discuss how soBGP can meet the first two goals in operational networks.

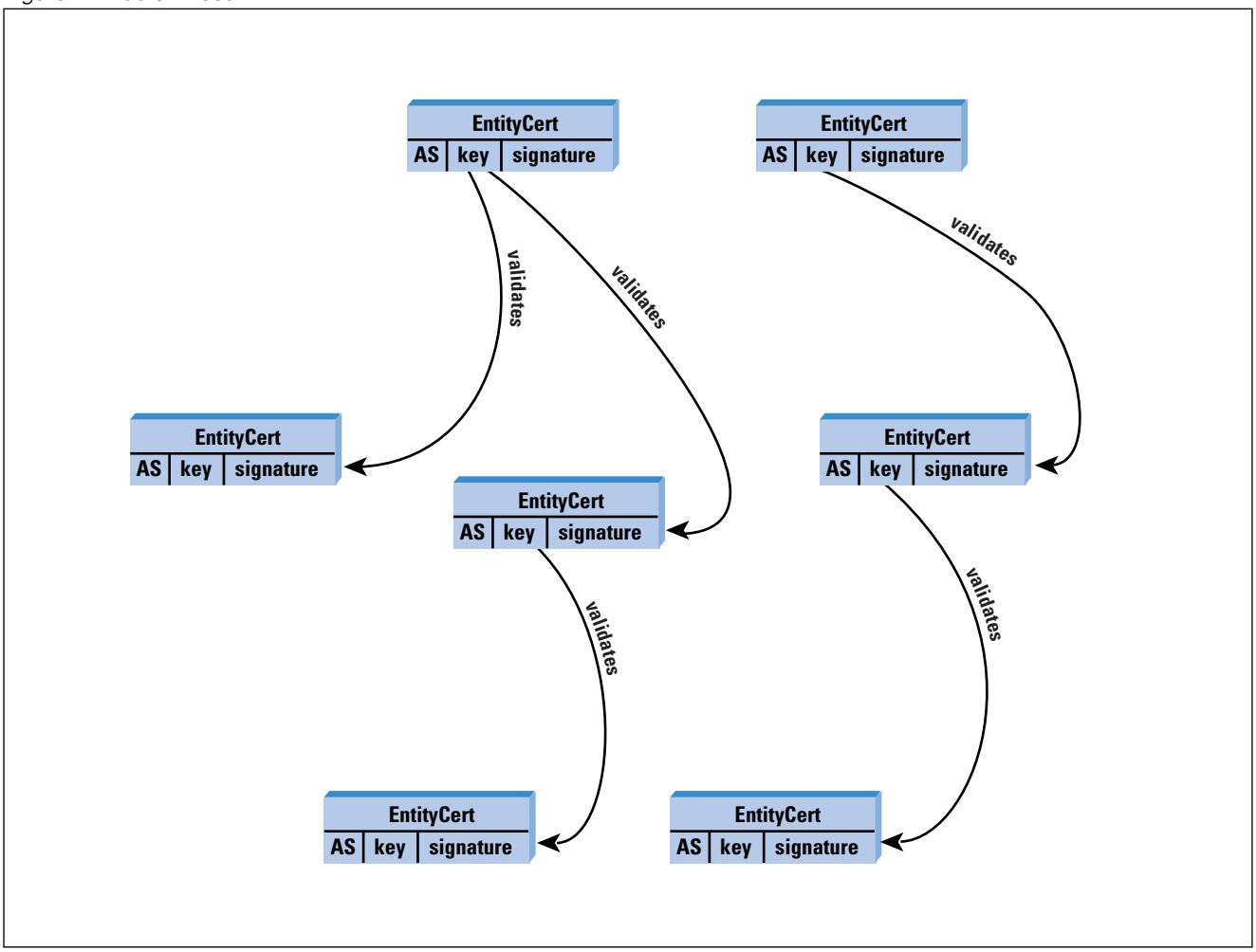
Begin at the Beginning: Who Are You?

The first step in securing anything is authentication; each participant must have some way of knowing who the other participants are, and what information they will be using to sign or encrypt their data. This is a classic problem in cryptography, called *key distribution*. There must be some way to receive keys used to sign or encrypt data, and then to validate that the keys received actually belong to the participant we believe they belong to.

This problem is addressed in soBGP using an *EntityCert,* which ties an AS number to a public key (or a set of public keys) corresponding to a private key the AS will be using to sign various other certificates. An EntityCert is defined in soBGP to be an X.509v3 certificate, similar to those used by *Transport Layer Security* (TLS) and *IP Security* (IPSec). The main problem we face when accepting an EntityCert is knowing whether or not the key carried within the certificate is actually the key of the advertising AS.

soBGP resolves this by requiring the EntityCert to be signed by a third party, validating that this AS actually belongs with this key. A small number of "root keys" distributed out of band could then be used to validate a set of advertised EntityCerts. These are used in turn to build up the database of known good ASm/key pairs in the system, allowing even more EntityCerts to be validated. Thus, EntityCerts can form a web of trust, built on the public keys of a small number of well-known entities, such as top-level backbone service providers, key authentication service providers (such as Verisign), and others.

Figure 1: Web of Trust

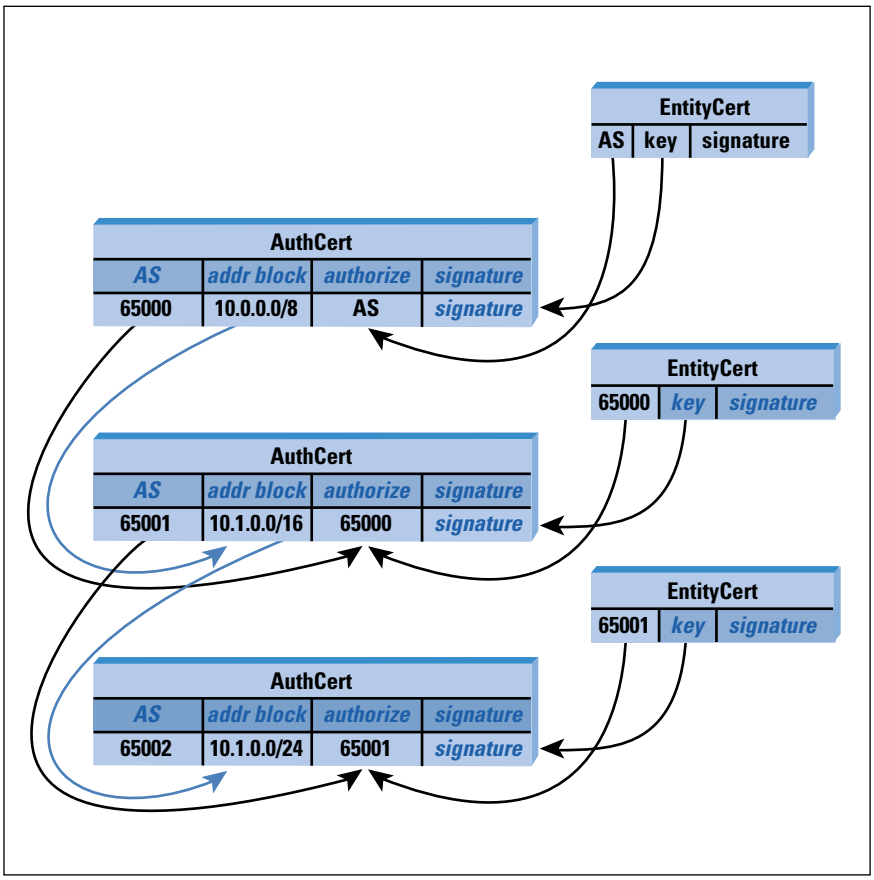


The key each AS distributes in its EntityCert is actually the public half of a private/public key pair. An AS would keep its private key entirely private, holding it on one highly secure device in its network (which is not even required to be online), and generating signatures for other certificates as needed. Only an AS public key is ever exposed in this way, so no special protection mechanisms (for example, tamper-resistant hardware) are required at any border to prevent private keys from being compromised.

The First Goal: Are You Authorized?

Now that we have distributed a public key per AS, we can build a certificate that will provide authorization for an AS to advertise a specific block of addresses. This authorization is provided through an *Authorization Certificate,* or *AuthCert*. An AuthCert ties an AS to a block of addresses that the AS may advertise, as Figure 2 illustrates.

Figure 2: Authorization Example



Starting at the top of the illustration, we find that some AS has authorized AS65000 to advertise prefixes within the block 10.0.0.0/8. The AuthCert is signed using the authorizing AS key. To delegate some part of this block of address space to another AS, AS65001, AS65000 builds an AuthCert tying 10.1.0.0/16 to AS65001. AS65001, in turn, suballocates a smaller part of this address space to AS65002, by building an AuthCert tying AS65002 to 10.1.1.0/24.

Any device receiving these three AuthCerts can check them by:

- Looking up the public key of the authorizer, and verifying the signature on the AuthCert
- Making certain the authorizer is permitted to advertise the address space it has suballocated this block of address space from

The device then builds a local table of address blocks and corresponding ASs authorized to advertise prefixes within those address blocks. Received updates can be checked against this database to verify authorization of the originating AS to advertise a prefix.

Blocks of address space are used here, rather than individual prefixes; an AuthCert can authorize an AS to advertise any number of prefixes within a block of addresses. This reduces the number of certificates within the system, thereby reducing overall cryptographic processing requirements. If a specific AS desires per-prefix authorization, it can build individual AuthCerts for each allocated prefix, rather than for blocks of address space.

Per-Prefix Policy

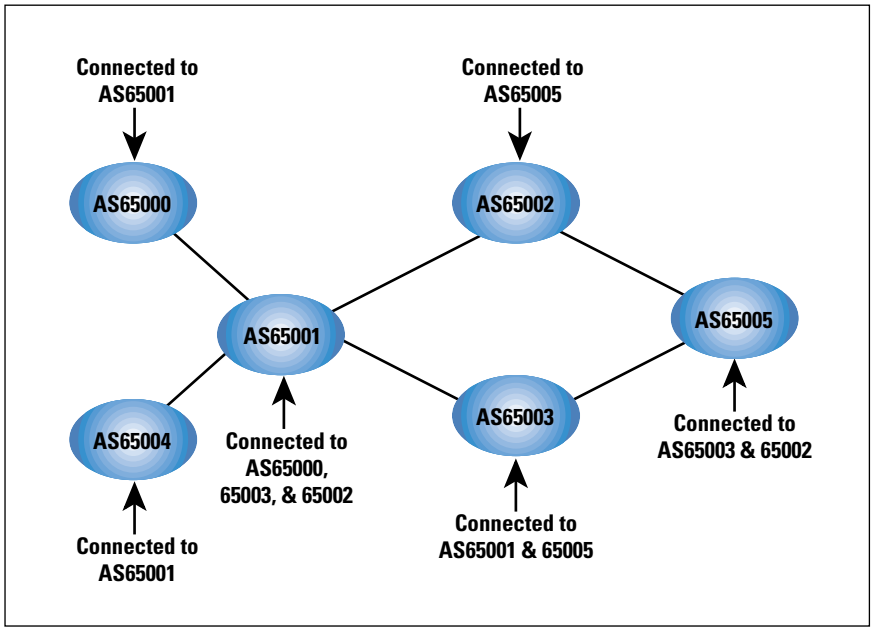
AuthCerts are not advertised as independent certificates within soBGP; instead, they are wrapped in a *PrefixPolicyCert*. PrefixPolicyCerts contain an AuthCert, a set of policies the originator would like to apply to prefixes advertised within this block of addresses, and a signature generated using the private key of the authorized AS. Policies that may be included in the PrefixPolicyCert include the longest prefix length allowed within the address block, and possibly other policies, such as a list of ASs that may not be or must be in the AS Path of routes to destinations within the address block.

In reality, the per-prefix policies available to the originator are limitless; the main problem is enforcing those policies when they are received by other ASs.

The Second Goal: Do You Really Have a Path?

Our second goal is to be able to verify that the advertiser of a given route actually has a path to the destination. This goal is met in soBGP by building a topology map of the paths of the entire internetwork. Each AS attached to the internetwork builds an *ASPolicyCert,* which contains, primarily, a list of its peers, and signed using the originator's private key. Using this list of transit peers, a map of the internetwork topology may be built, as Figure 3 illustrates.

Figure 3: Connectivity Graph Example



If AS65005 receives an update from AS65002, claiming it can reach a destination in AS65000 through the path {65002, 65001, 65000}, it can:

- Check to make certain AS65002 claims to be connected to AS65001 in its ASPolicyCert, and that AS65001 claims to be connected to AS65002 in its ASPolicyCert
- Check to make certain AS65001 claims to be connected to AS65000 in its ASPolicyCert, and that AS65000 claims to be connected to AS65001 in its ASPolicyCert

If, for instance, AS65002 claims a path to a destination inside AS65000 through the path (65002, 65000), AS65002 would be able to discover that the path is invalid, because AS65000 does not claim to be connected to AS65002. This simple two-way connectivity check along a graph can be mixed with various policy statements—stating a specific peer is not a transit, not advertising certain peers, etc.—to provide a much wider range of policies than AS Path-based methods.

Transporting Certificates

One of the primary problems any security system such as soBGP is going to face is transporting security information through the internetwork. We would like to make certain we do not rely on the routing system to provide information about the security of the routing system. In other words, we would not like to rely on unsecured routing information in order to reach a server providing the information required to secure the path to the server itself.

soBGP resolves this by proposing to advertise certificates in much the same way as routing information is propagated today—through an interdomain protocol. Currently the soBGP drafts specify a new type of BGP message, the SECURITY message, which can be used to transport the required certificates, the EntityCert, the PrefixPolicyCert, and the ASPolicyCert, throughout an internetwork. Other methods of transporting data such as these certificates throughout an internetwork are currently being pursued by the IETF; if other methods are offered, soBGP could transport certificates across any such distribution mechanism.

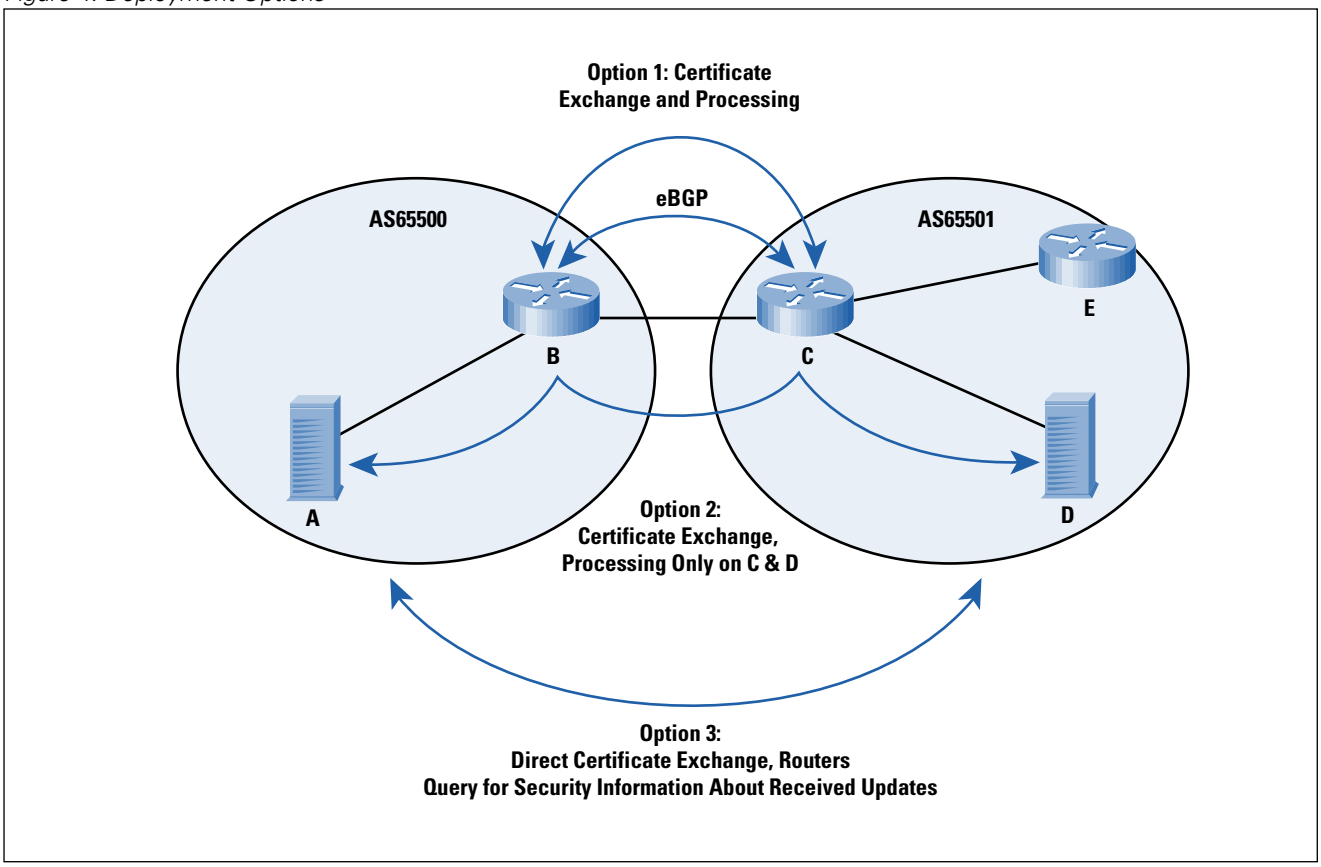
Deployment

Finally, we come to the hardest problem any routing security system is going to face: actually getting it operating in the field, with useful results, with a minimum of equipment changes, and a minimum number of participants. Here, soBGP provides a wide variety of options, primarily because it is not transport-dependent, nor dependent on a yet-to-be constructed centralized set of servers.

Although deployment options abound, here we discuss three, just to show the range of options available. Figure 4 illustrates these options.

The first option shown in this network is direct certificate exchange and processing between border routers. With this option, routers that are capable of the cryptographic processing required to validate received certificates exchange certificates with their peers in other ASs (just as they exchange routing information today), process those certificates, and build local databases from which they perform security checks on received updates.

Figure 4: Deployment Options



Although it may appear that processing, in this situation, would be extensive, it is actually possible to spread the processing required out among the border routers in a large AS. For instance, each certificate that router C receives and processes can be subsequently sent over an encrypted link to Router E. Router E could treat these certificates as though they had been validated locally, because they are received across an encrypted link from a trusted peer within the same administrative domain. Thus, only the edge router that has learned a certificate would actually process the certificate. This spreads the processing along all the edges in the AS.

A second option is for the edge routers, B and C, to exchange the certificates, but not process them. Instead, each edge router would relay the not-yet-validated certificates to internal servers A and D, respectively, thereby validating the certificates by performing the necessary cryptographic operations. As the border routers receive updates, they can query the server about the validity of each update, and take action based on the reply received.

Finally, it is possible for the servers to exchange certificates directly, over a multihop session. Servers A and D would then process the certificates, and the border routers, B and C, would query these servers to determine if received updates are valid or invalid.

Summary

Through this short survey of soBGP, we have shown it to be a flexible, moderately lightweight, yet strong system for validating the information carried through BGP in a large internetwork. It has low overhead processing requirements and very flexible deployment options, but no reliance on centralized servers. We are currently working to develop prototypes of soBGP on several platforms, to show how the technology will work on a wide range of devices.

For more information on soBGP, refer to:
`ftp://ftp-eng.cisco.com/sobgp/index.html`

You will find the most recent versions of the drafts, several slide shows, and other information about soBGP at this site.

RUSS WHITE works for Cisco Systems in the Routing Protocols Deployment and Architecture (DNA) team in Research Triangle Park, North Carolina. He has worked in the Cisco Technical Assistance Center (TAC) and Escalation Team in the past, has coauthored several books on routing protocols, including *Advanced IP Network Design, IS–IS for IP Networks,* and *Inside Cisco IOS® Software Architecture*. He is currently in the process of publishing a book on BGP deployment, and is the cochair of the Routing Protocols Security Working Group within the IETF. E-mail: **`riw@cisco.com`**

Trends in Viruses and Worms

by Thomas M. Chen, Southern Methodist University

The modern computer *virus* was conceived and demonstrated by Fred Cohen in 1983. Like biological viruses, computer viruses reproduce by attaching to a normal program or document and taking over control of the execution of that program to infect other programs. Early viruses could spread slowly mostly by floppies (such as the 1986 *Brain* virus), but the Internet has made it much easier for viruses to move among computers and spread rapidly. Networks have created a fertile environment for worms, which are related to viruses in their ability to self-replicate but are not attached to other programs. Worms are particularly worrisome as standalone automated programs designed to exploit the network to seek out vulnerable computers. The term *worm* was originated by John Shoch and Jon Hupp during their experiments on mobile software at Xerox PARC in 1979, inspired by the network-based *tapeworm* monster in John Brunner's novel, *The Shockwave Rider*[1]. Shoch and Hupp thought of worms as multi-segmented programs distributed across networked computers.

The Internet increases the vulnerability of all interconnected machines by making it easier for malicious programs to travel between computers by themselves. Recent virus and worm outbreaks, such as the *Blaster* worm in August 2003 and the SQL *Sapphire/Slammer* worm in January 2003, have demonstrated that networked computers continue to be vulnerable to new attacks despite the widespread deployment of antivirus software and firewalls. Indeed, a review of the history of viruses and worms shows that they have continually grown in sophistication over the years. This article highlights a series of significant past innovations in virus and worm technology. The purpose is to show that viruses and worms continue to pose a major risk today and most likely into the future as their creators persist in seeking ways to exploit security weaknesses in networked systems.

Stealth

The earliest viruses attempted to hide evidence of their presence, a trend that continues to today. The 1986 DOS-based *Brain* virus hid itself in memory by simulating all of the DOS system calls that normally detect viruses, causing them to return information that gave the appearance that the virus was not there.

The 2001 *Lion* worm installed a rootkit called *t0rn*, which is designed to make the actions of the worm harder to detect through numerous system modifications to deceive *syslogd* from properly capturing system events (*syslogd* is often used to detect worm activity)[2]. More recently, viruses and worms have attempted to hide by actively attacking antivirus software on the infected computer (refer to the section "Armoring").

Social Engineering

The 1987 *Christma Exec* virus was an early example of social engineering, spreading by e-mail among IBM mainframes. An arriving message tricks the user into executing the virus by promising to draw a Christmas tree graphic. The virus does produce a Christmas card graphic on the computer display (drawn using a scripting language called *Rexx*) but sends a copy of itself in the user's name to that user's list of outgoing mail recipients. The recipients believe the e-mail is from the user, so they are more likely to open the e-mail.

Social engineering continues to be common practice in today's viruses and worms, particularly those spread by e-mail. In January 1999, the *Happy99/Ska* worm/Trojan horse hybrid spread by e-mail with an attachment called **`Happy99.exe`**[3]. When the attachment was executed, it displayed fireworks on the screen to commemorate New Year's Day, but secretly modified the **`WSOCK32.DLL`** file (the main Windows file for Internet communications) with a Trojan horse program that allowed the worm to insert itself into the Internet communications process. Every e-mail sent by the user generated a second copy without any text but carried the worm to the same recipients.

The 1999 *PrettyPark* worm propagated as an e-mail attachment called **`Pretty Park.exe`**. The attachment is not explained, but it bears the icon of a character from the television show, *South Park*. If executed, it installs itself into the Windows System folder and modifies the Registry to ensure that it runs whenever any **`.EXE`** program is executed. In addition, the worm e-mails itself to addresses found in the Windows Address Book. It also mails some private system data and passwords to certain *Internet Relay Chat* (IRC) servers. Reportedly, the worm also installs a backdoor to allow a remote machine to create and remove directories, and send, receive, and execute files.

In February 2001, the *Anna Kournikova* virus demonstrated social engineering again, pretending to carry a JPG picture of the tennis player. If executed, the virus e-mails a copy of itself to all addresses in the Outlook address book.

In March 2002, the *Gibe* worm spread as an attachment in an e-mail disguised as a Microsoft security bulletin and patch. The text claimed that the attachment was a Microsoft security patch for Outlook and Internet Explorer. If the attachment is executed, it displays dialog boxes that appear to be patching the system, but a backdoor is secretly installed on the system.

Macro Viruses

The *Concept* virus was the first macro virus, written for Word for Windows 95. The vast majority of macro viruses are targeted to Microsoft Office documents that save macro code within the body of documents. Macro viruses have the advantages of being easy to write and independent of computing platform. However, macro viruses are no longer widespread after people have become more cautious about using the Office macro feature.

Mass E-Mailers

In March 1999, the *Melissa* macro virus spread quickly to 100,000 hosts around the world in three days, setting a new record and shutting down e-mail for many organizations using Microsoft Exchange Server[4]. It began as a newsgroup posting promising account names and passwords for erotic Web sites. However, the downloaded Word document actually contained a macro that used the functions of Microsoft Word and the Microsoft Outlook e-mail program to propagate. Up to that time, it was widely believed that a computer could not become infected with a virus just by opening e-mail. When the macro is executed in Word, it first checks whether the installed version of Word is infectable. If it is, it reduces the security setting on Word to prevent it from displaying any warnings about macro content. Next, the virus looks for a certain Registry key containing the word "Kwyjibo" (apparently from an episode of the television show, *The Simpsons*). In the absence of this key, the virus launches Outlook and sends itself to 50 recipients found in the address book. Additionally, it infects the Word **`NORMAL.DOT`** template using the Microsoft *Visual Basic for Applications* (VBA) macro auto-execute feature. Any Word document saved from the template would carry the virus.

In June 1999, the *ExploreZip* worm appeared to be a WinZip file attached to e-mail but was not really a zipped file[5]. If executed, it appears to display an error message, but the worm secretly copies itself into the Windows Systems directory or loads itself into the Registry. It sends itself via e-mail using Outlook or Exchange to recipients found in unread messages in the inbox. It monitors all incoming messages and replies to the sender with a copy of itself.

In May 2000, the fast-spreading *Love Letter* worm demonstrated a social engineering attack[6]. It propagated as an e-mail message with the subject "I love you" and text that encourages the recipient to read the attachment. The attachment is a Visual Basic script that could be executed with Windows Script Host (present if the computer has Windows 98, Windows 2000, Internet Explorer 5, or Outlook 5). Upon execution, the worm installs copies of itself into the Windows System directory and modifies the Registry to ensure that the files are run when the computer starts up. The worm also infects various types of files (for example, **`.VBS`**, **`.JPG`**, **`.MP3`**, etc.) on local drives and networked shared directories. If Outlook is installed, the worm e-mails copies of itself to addresses found in the address book. In addition, the worm makes a connection to IRC and sends a copy of itself to anyone who joins the IRC channel. The worm has a password-stealing feature that changes the startup URL in Internet Explorer to a Website in Asia. The Website downloads a Trojan horse designed to collect various passwords from the computer.

In 2002, 90 percent of the known viruses were mass e-mailers. Two of the most prevalent ones, *Bugbear* and *Klez,* began a trend of carrying their own *Simple Mail Transfer Protocol* (SMTP) engines. Although e-mail continues to be the most common infection vector, recent worms have been exploring new vectors (see the section "New Infection Vectors").

In addition, mail servers are becoming more powerful in their capabilities to detect and filter malicious code. For these reasons, mass e-mailing may decline as an infection vector for future viruses.

Polymorphism

Polymorphism is based on the simpler idea of encryption, which makes a virus harder to detect by antivirus software scanning for a unique virus signature (byte pattern). Encryption attempts to hide a recognizable signature by scrambling the virus body. To be executable, the encrypted virus is prepended with a decryption routine and encryption key. However, encryption is not effective because the decryption routine remains the same from generation to generation, although the key can change, scrambling the virus body differently. Antivirus scanners can detect a sequence of bytes identifying a specific decryption scheme.

Polymorphic viruses permute continuously to avoid detection by antivirus scanning[7]. The earliest polymorphic virus might have been a virus found in Europe in 1989. This virus replicated by inserting a pseudorandom number of extra bytes into the decryption algorithm, preventing any common sequence of more than a few bytes between two successive infections. Polymorphism became practical when a well-known hacker, *Dark Avenger,* developed a user-friendly *Mutation Engine* program to provide any virus with variable encryption. With a static signature so small, the risk of false positives by antivirus scanners became very high. Other hackers soon followed with their own versions of so-called mutation engines. The 1995 *Pathogen* and *Queeg* viruses were polymorphic DOS file-infecting viruses produced by Black Baron's *Simulated Metamorphic Encryption enGine* (SMEG)[7].

Blended Attacks

The famous 1988 *Morris* worm was the first to use a combination of attacks (or blended attacks) to spread quickly to 6000 UNIX computers in a few hours (10 percent of the Internet at that time)[8].

- It captured the password file and ran a password-guessing program on it using a dictionary of common words.
- It exploited the debug option in the UNIX *sendmail* program, allowing it to transfer a copy of itself.
- It carried out a buffer overflow attack through a vulnerability in the UNIX *fingerd* program.

In May 2001, the *Sadmind/IIS* worm spread by targeting two separate vulnerabilities on two different operating systems. It first exploited a buffer overflow vulnerability in Sun Solaris systems and installed software to carry out an attack to compromise Microsoft *Internet Information Services* (IIS) Web servers.

The July 2001 *Sircam* worm uses two ways to propagate. First, it e-mails itself as an attachment using its own SMTP engine, and if the attachment is executed, e-mails a copy of itself to addresses found in the Windows address book. Second, it spreads by infection of unprotected network shares.

In September 2001, *Nimda* raised new alarms by using five different ways to spread to 450,000 hosts within the first 12 hours[9]. *Nimda* seemed to signal a new level of worm sophistication.

- It found e-mail addresses from the computer Web cache and default *Messaging Application Programming Interface* (MAPI) mailbox. It sent itself by e-mail with random subjects and an attachment named **`readme.exe`**. If the target system supported the automatic execution of embedded MIME types, the attached worm would be automatically executed and infect the target.
- It infected Microsoft IIS Web servers, selected at random, through a buffer overflow attack called a *unicode* Web traversal exploit.
- It copied itself across open network shares. On an infected server, the worm wrote *Multipurpose Internet Mail Extensions* (MIME)-encoded copies of itself to every directory, including network shares.
- It added JavaScript to Web pages to infect any Web browsers going to that Website.
- It looked for backdoors left by previous *Code Red II* and *Sadmind* worms.

Armoring

In November 2002, the *Winevar* worm was an example of an "armored" worm that contained special code designed to disable antivirus software using a list of keywords to scan memory to recognize and stop antivirus processes and scan hard drives to delete associated files[10].

Klez and *Bugbear* are recent examples of worms that attack antivirus software by stopping active processes and deleting registry keys and database files used by popular antivirus programs. The 2003 *Fizzer* and *Lirva* worms also attempt to disable antivirus software.

Dynamic Software Updates

In October 2000, the *Hybris* worm propagated as an e-mail attachment[11]. It connected to the **`alt.comp.virus`** newsgroup to receive encrypted plug-ins (code updates). The method is sophisticated and potentially very dangerous, because the worm payload (destructive capability) can be modified dynamically.

The 2003 *Lirva* worm attempted to connect to a Website on **`web.host.kz`** to download BackOrifice, a notorious remote-access software package that gives complete control to a remote attacker. It also attempted to download another unknown file that was not found on the Website.

This technique was given an interesting twist by the *Welchia* or *Nachi* worm, which began spreading on August 18, 2003, soon after the *Blaster* worm. Apparently, its creator intended *Welchia* as a "good" worm to remove *Blaster*. It attempted to download and install a fix for *Blaster* from a Microsoft Website.

New Infection Vectors

The Linux *Slapper* worm, appearing in September 2002, was among the first to exploit *peer-to-peer* (P2P) technology[12]. It spread to Linux computers by exploiting the long *Secure Sockets Layer 2* (SSL2) key argument buffer overflow in the *libssl* library, used by the *mod_ssl* module of the Apache 1.3 Web server. When the worm infects a new machine, it binds to *User Datagram Protocol* (UDP) port 2002 and becomes part of a P2P network. The parent of the worm on the attacking machine sends to its offspring the list of all hosts on the P2P network and broadcasts the address of the new worm on the network. Then periodic updates to the host list are exchanged between machines on the network. The new worm also scans the network for other vulnerable machines, sweeping randomly chosen class B networks.

In March 2003, the *AimVen* worm spread by the *America OnLine Instant Messager* (AIM) by modifying the AIM program. Whenever an **.EXE** file is sent through AIM, the worm overwrites the file with a copy of itself.

The *Fizzer* worm discovered in May 2003 is a mass e-mailer that includes its own SMTP engine like *Klez* and *Bugbear*. It also tries to spread via *KaZaa*, a popular P2P file-sharing application, and shared directories.

The 2003 *Lirva* worm, named after the singer, Avril Lavigne, is a mass e-mailer taking advantage of the same MIME header exploit as *Badtrans* and *Klez*, but also tries to spread by IRC, "I seek You" (ICQ), *KaZaa,* and open network shares[13].

Data-Stealing Payloads

Most fast-spreading worms in the past have not carried destructive payloads. Instead, they have tended to appear to be proof-of-concepts to demonstrate a particular security weakness. Some worms, though, such as *Code Red,* have installed *Denial-of-Service* (DoS) agents or backdoors on infected machines. Recently worms have begun to carry keyloggers and password-stealing Trojans in their payloads.

The 2003 *Fizzer* worm includes a keystroke logging Trojan horse that stores the data in an encrypted file. It establishes its own accounts on IRC and AIM to wait for instructions from the virus writer, who could conceivably fetch the keystrokes data.

The 2003 *Lirva* worm e-mails cached Windows dialup networking passwords to the virus writer, and e-mail random **.TXT** and **.DOC** files to various addresses.

Bugbear installs a keystroke logging tool into the Windows System folder that e-mails the keystrokes data to preprogrammed addresses[14]. It listens on port 36794 for commands from a remote hacker.

Fast and Furious Worms

A particularly worrisome new trend is extremely fast worms targeted to specific (usually Windows-related) vulnerabilities that might saturate their target population within a few hours or even less than an hour. These worms tend to be simpler and targeted to single rather than multiple vulnerabilities, in order to be highly efficient in their probing for other vulnerable machines.

The first example might be the *Code Red* worm, which actually appeared in three different versions[15]. The first version of *Code Red I* appeared on July 12, 2001, targeted to a buffer overflow vulnerability in Microsoft IIS Web servers. However, a programming error in its pseudorandom address generator caused each worm copy to probe the same set of IP addresses and prevented the worm from spreading quickly. A week later on July 19, a second version of *Code Red I* with the programming error apparently fixed was able to infect more than 359,000 servers within 14 hours. At its peak, the worm was infecting 2000 hosts every minute. A more complex and dangerous *Code Red II* targeted to the same IIS vulnerability appeared on August 4.

More recently, the *Structured Query Language* (SQL) *Sapphire/ Slammer* worm appeared on January 25, 2003, targeted to Microsoft SQL Server machines not running *Service Pack 3* (SP3), such as SQL Server 2000 and *Microsoft Desktop Engine* (MSDE) 2000[16]. It reportedly infected 90 percent of vulnerable hosts within 10 minutes (about 120,000 servers)[17]. The spreading rate was surprisingly fast and resulted in DoS effects (network outages and high packet loss) due to traffic overloading servers and routers. In the first minute, the infection doubled every 8.5 seconds, and hit a peak scanning rate of 55,000,000 scans per second after only 3 minutes. In comparison, *Code Red* infection doubled in 37 minutes (slower but infected more machines). *Slammer* was able to spread so quickly because it appeared to be designed simply for efficient replication. The worm carried no payload and consisted of a single 404-byte UDP packet (including 376 bytes for the worm) that could be sent without having to wait for responses from targeted machines. In contrast, *Code Red* was about 4000 bytes and *Nimda* was 60,000 bytes, and their scanning depended on the time to establish TCP connections to targeted machines. The *Slammer* worm was much more efficient, simply generating copies of itself at the full rate of the infected machine.

Latest Developments

The week of August 12–19, 2003, has been called the worst week for worms in history, seeing *MS Blaster, Welchia* (or *Nachi*), and *Sobig.F* in quick succession. *MS Blaster* or *LovSan* was another fast worm, which appeared on August 12, 2003, targeted to a *Windows Distributed Component Object Model* (DCOM) *Remote Procedure Call* (RPC) vulnerability announced on July 16, 2003[18]. The worm probes for a DCOM interface with RPC listening on TCP port 135 on Windows XP and Windows 2000 PCs. Through a buffer overflow attack, the worm causes the target machine to start a remote shell on port 4444 and send a notification to the attacking machine on UDP port 69.

A *Trivial File Transfer Protocol* (TFTP) "get" command is then sent to port 4444, causing the target machine to fetch a copy of the worm as the file **`MSBLAST.EXE`**. In addition to a message against Microsoft, the worm payload carries a DoS agent (using TCP SYN flood) targeted to the Microsoft Website **`windowsupdate.com`** on August 16, 2003. Although *Blaster* has reportedly infected about 400,000 systems, experts reported that the worm did not achieve near its potential spreading rate because of novice programming.

Six days later on August 18, 2003, the apparently well-intended *Welchia* or *Nachi* worm spread by exploiting the same RPC DCOM vulnerability as *Blaster*. It attempted to remove *Blaster* from infected computers and download a security patch from a Microsoft Website to repair the RPC DCOM vulnerability. Unfortunately, its scanning resulted in a DoS effect on some networks, such as Air Canada's check-in system and the U.S. Navy and Marine Corps computers.

The very fast *Sobig.F* worm appeared on the next day, August 19, 2003, only seven days after *Blaster*[19]. The original *Sobig.A* version was discovered in January 2003, and apparently underwent a series of revisions until the most successful *Sobig.F* variant. Similar to earlier variants, *Sobig.F* spreads among Windows machines by e-mail with various subject lines and attachment names, using its own SMTP engine. The worm size is about 73 kilobytes with a few bytes of garbage attached to the end to evade antivirus scanners. It works well because it grabs e-mail addresses from a variety of different types of files on the infected computer and secretly e-mails itself to all of them, pretending to be sent from one of the addresses. At its peak, *Sobig.F* accounted for 1 in every 17 messages, and reportedly produced over 1 million copies of itself within the first 24 hours. Interestingly, the worm was programmed to stop spreading on September 10, 2003, suggesting that the worm was intended as a proof-of-concept. This is supported by the absence of a destructive payload, although the worm is programmed with the capability to download and execute arbitrary files to infected computers. The downloading is triggered on specific times and weekdays, which are obtained via one of several *Network Time Protocol* (NTP) servers. The worm sends a UDP probe to port 8998 on one of several preprogrammed servers, which responds with a URL for the worm to download. The worm also starts to listen on UDP ports 995–999 for incoming messages, presumably instructions from the creator.

Conclusions

Why does the Internet remain vulnerable to large-scale worm outbreaks? Since at least 1983, the Internet community has understood the risks and mechanics of viruses. The 1988 Morris worm taught the community to be watchful for potentially dangerous worms. Over the years, a variety of antivirus software, firewalls, intrusion detection systems, and other security equipment have been installed. Moreover, the *Computer Emergency Response Team* (CERT) at CMU was established as the first computer security incident response team, which later joined an expansive global coalition of security incident response teams called the *Forum of Incident Response and Security Teams* (FIRST)[20].

Despite our knowledge and infrastructure defenses, many viruses and worms have broken out regularly in the Internet over the years. By some reports, 5 to 15 new viruses and worms are released every day, although a fraction of that number are not released in the wild and most do not spread well. Still, fast-spreading viruses and worms continue to appear with regularity. Outbreaks have become so commonplace that most organizations have come to view them as a routine cost of operation.

The problem is sometimes portrayed as a perpetual struggle between virus writers who keep innovating (as described here) and the antivirus industry, which tries to keep up. However, the problem is actually larger, involving the entire computer industry. Viruses and worms are successful because computers have security vulnerabilities that can be exploited. Clearly, the Internet itself is simply serving its purpose of interconnecting computer systems. The security vulnerabilities exist in the host end systems. Security vulnerabilities continue to exist for many reasons. First, software is often written in an unsecure manner, for example, vulnerable to buffer overflow attacks that are commonly used by worms. Buffer overflow attacks have been widely known since 1995, but this type of vulnerability continues to be found very often (on every operating system.) Second, when vulnerabilities are announced with corresponding software patches, many people are slow to apply patches to their computer for various practical reasons. Weakly protected computers can be compromised, putting the entire community at risk, including secured computers that can still be impacted by the traffic effects of a worm outbreak.

However, there is reason to be hopeful for a solution. Fortunately, worms typically have a weakness of exploiting vulnerabilities that have been known for some time. Worm writers do not invent new exploits for the simple reason that they want to ensure that their worm will spread after it is released. For example, the *Code Red I* worm took advantage of a buffer overflow vulnerability in Microsoft IIS servers that had been known for a month. The *Nimda* worm exploited a unicode Web traversal vulnerability in Microsoft IIS servers that was published a year earlier. The SQL *Slammer/Sapphire* worm exploited a buffer overflow vulnerability in Microsoft SQL servers that had been known for six months. The recent *Blaster* worm exploited a Windows DCOM RPC vulnerability announced two months earlier. Watching for probing activity attempting to exploit known vulnerabilities could help detect and block worm outbreaks at an early stage. Ideas for automatic detection and quarantine of new epidemics is attracting research[21].

Aside from technological considerations, an important issue is accountability. The most obvious parties to hold liable are the virus creators, but it has been observed many times that few virus writers have been prosecuted, and sentences have tended to be light. The author of the 1988 Internet worm, Robert Morris, was sentenced to three years of probation, 400 hours of community service, and a $10,000 fine.

Chen Ing-hau was arrested in Taiwan for the 1998 *Chernobyl* virus, but he was released when no official complaint was filed. Onel de Guzman was arrested for writing the 2000 *LoveLetter* virus, which resulted in $7 billion of damages, but he was released because of the lack of relevant laws in the Philippines. Jan De Wit was sentenced for the 2001 *Anna Kournikova* virus to 150 hours of community service. David L. Smith, creator of the 1999 *Melissa* that caused at least $80 million of damages, was sentenced to 20 months of custodial service and a $7500 fine.

It is notoriously difficult to trace a virus or worm to its creator from analysis of the code, unless inadvertent clues are left in the code. In addition, cases are difficult to prosecute, and malicious intention (as opposed to just recklessness) is difficult to prove. Moreover, long prison sentences have been perceived as overly harsh for arrested virus creators, who have tended to be teenagers and university students. In addition, in the absence of a serious legal deterrent, the general perception persists that virus creators can easily avoid the legal consequences of their actions. Perhaps to address this problem, authorities have been diligently investigating the creators of *Blaster* and *Sobig*. So far, a teenager, Jeffrey Lee Parson, has been arrested for writing the *Blaster.B* variant, a slight modification of the original *Blaster*. Soon afterward, Dan Dumitru Ciobanu was arrested in Romania for writing the *Blaster.F* variant.

Some have argued wishfully that software vendors should be held financially liable for damages resulting from the security vulnerabilities in their products. The assumption is that accountability would increase motivation to write and sell more secure software, a solution that would result in a less inviting environment for viruses and worms. So far, software vendors have managed to acknowledge their role but avoid accountability.

References

[1] J. Shoch and J. Hupp, "The 'worm' programs—early experience with a distributed computation," *Communications of ACM,* Volume 25, pp. 172–180, March 1982.

[2] A. Kasarda, "The Lion worm: king of the jungle?" SANS reading room, **http://www.sans.org/rr**

[3] CERT incident note CA-1999-02, "Happy99.exe trojan horse," **http://www.cert.org/incident_notes/IN-99-02.html**

[4] CERT advisory CA-1999-04, "Melissa macro virus," **http://www.cert.org/advisories/CA-1999-04.html**

[5] CERT advisory CA-1999-06, "ExploreZip trojan horse program," **http://www.cert.org/advisories/CA-1999-06.html**

[6] CERT advisory CA-2000-04, "Love letter worm," **http://www.cert.org/advisories/CA-2000-04.html**

[7] D. Harley, R. Slade, and R. Gattiker, *Viruses Revealed,* Osborne/McGraw-Hill, 2001.

[8] E. Spafford, "The Internet worm program: an analysis," *ACM Computer Communications Review,* Volume 19, pp. 17–57, January 1989.

[9] CERT advisory CA-2001-26, "Nimda worm,"
http://www.cert.org/advisories/CA-2001-26.html

[10] Virus Bulletin, "W32/WineVar,"
http://www.virusbtn.com/resources/viruses/winevar.xml

[11] CERT incident note IN-2001-02, "Open mail relays used to deliver Hybris worm,"
http://www.cert.org/incident_notes/IN-2001-02.html

[12] F-Secure, "F-Secure virus descriptions: Slapper,"
http://www.f-secure.com/v-descs/slapper.shtml

[13] Symantec Security Response, "W32.lirva.C@mm,"
http://securityresponse.symantec.com/avcenter/venc/data/w32.lirva.c@mm .html

[14] Sophos, "W32/Bugbear-A,"
http://www.sophos.com/virusinfo/analyses/w32bugbeara.html

[15] H. Berghel, "The Code Red worm," *Communications of ACM,* Volume 44, pp. 15–19, December 2001.

[16] CERT advisory CA-2003-04, "MS-SQL server worm,"
http://www.cert.org/advisories/CA-2003-04.html

[17] D. Moore, et al., "The spread of the Sapphire/Slammer worm,"
http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html

[18] CERT advisory CA-2003-20, "W32/Blaster worm," Aug. 11, 2003,
http://www.cert.org/advisories/CA-2003-20.html

[19] Symantec Security Response, "W32.Sobig.F@mm,"
http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm .html

[20] Forum of Incident Response and Security Teams (FIRST),
http://www.first.org

[21] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: requirements for containing self-propagating code," IEEE Infocom 2003, San Francisco, April 2003.

THOMAS M. CHEN holds BS and MS degrees in electrical engineering from MIT, and a PhD in electrical engineering from the University of California, Berkeley. From 1989 to 1997, he worked on ATM networking research at GTE Laboratories (now Verizon). He is currently an Associate Professor in the Department of Electrical Engineering at SMU in Dallas, Texas. He is the associate editor-in-chief of *IEEE Communications Magazine,* a senior editor of *IEEE Network,* an associate editor of *ACM Transactions on Internet Technology,* and founding editor of *IEEE Communications Surveys.* He is the coauthor of *ATM Switching Systems* (Artech House, 1995). E-mail: **tchen@engr.smu.edu**

IPv6 Behind the Wall

by Jim Bound

IPv6 has technology advantages over IPv4, and most of them will not be seen by the end user any more than users see features added to other extensions to the Internet Protocol suite, sensors on their automobiles, or from any core technology evolution. This article focuses on three of those IPv6 technology advantages "Behind the Wall."

An essential catalyst for the Next-Generation Internet is the *Internet Protocol Version 6* (IPv6), which will provide an evolution to a more pervasive use of the Internet and networking in general. The current Internet, using IPv4, is insufficient to support the business and operational preconditions for peer-to-peer applications and security, billions of mobile devices, sensor networks, and the requisite distributed computing infrastructure to support a mobile society. The "band aids" applied to permit the current Internet to keep it operating has created additional operational costs and reduced operational capabilities for users and networks.

This article is an IPv6 Forum (**www.ipv6forum.com**) statement of the technology advantages of IPv6.

IPv6 Supports End-to-End Applications and Security

There are several schools of thought and opinions on the issue of address space and all project different results, depending on one's mathematical view and philosophy regarding use models. There is also the effect of disruptive technology, which can make moot any projections of IPv4 address space. In that sense, rationing is justified and intelligent. The IPv6 Forum believes we already are experiencing the initial quake of disruptive technology, and that there is a need for users and markets to evolve further with a basic tenet that end-to-end applications and security are a priori for that evolution to begin. The IPv6 Forum believes that *Network Address Translatio*n (NAT) is about control, but that control comes at a cost of the freedom to use peer-to-peer computing over client to server-only computing.

Two users on the Internet today generally cannot each initiate peer-to-peer communications with each other because their location and identity are not available to each other from two disparate networks. In addition, security between them must trust a third party, and absolute private communications is impossible. The reason is that the Internet has evolved so that users are generally behind NATs that preclude peer-to-peer communications, or the exchange of private security credentials. Some will say this affords users security on the Internet. Although NAT does provide a denial-of-service perimeter, it also provides a denial of service to a direct trust relationship between peers. IPv6 is the only way to have peer-to-peer security for the Next-Generation Internet at a reasonable cost and a true privacy trust model on the Internet.

In the field of network computer science when engineers and architects implement translation functions in a solution, a cost is incurred that would not exist without translation. This is due to the need to keep *state* before, during, and after the translation. In software engineering terminology, these *state machines* add time and space costs to the entire operation. In addition, a NAT box is a single point of failure, because it is the only point on the network where a user can exit or enter when translation exists. Translation also does not permit the use of all functions possible without translation because too many participants need to know the mappings, and each function requires a separate state to be maintained, and the time + space costs increase exponentially. The time + space costs of NAT to keep the Internet operational have been passed on to every part of the current Internet business, consumer, and government market sectors, and cannot even support the original functions of the Internet before NAT. The current Internet has no hope of supporting the functions of the Next-Generation Internet required or of offering a solution to the great digital divide that exists currently and is increasing daily.

The good news is that IPv6 is evolving, early adopter deployment has begun, and vendors have delivered initial IPv6 products to the market. IPv6 will not require NAT, and the infrastructure supports a stateless architecture for the Internet, using statefull properties only where they can be used without a translation attribute or policy. IPv6 inherently supports mobile communications, billions of devices, and sensor networks that will be pervasive at a reasonable cost and provide the option to eliminate the digital divide within the current Internet.

IPv6 Supports a Stateless Node Discovery Architecture

A Next-Generation Internet base technology advantage for mobile user devices, ad hoc networks, mobile network providers, and generally for all users is the *Stateless Node Discovery Architecture* inherent within IPv6.

IPv6 nodes can discover each other and form IPv6 addresses to communicate on a network using what is called *Neighbor Discovery* and *Stateless Autoconfiguration*. IPv6 supports an extensible stateless node discovery paradigm, which provides the following features:

- Discover presence of nodes on the network
- Discover Datalink Layer nodes on the network
- Discover routers on the network
- Discover link configuration parameters on the network

These features permit an IPv6 node to obtain and maintain information about the accessibility of another node on the network for communications. Node Discovery is the predecessor to the node obtaining an address from IPv6 autoconfiguration. This core IPv6 technology framework also permits nodes to communicate on networks where there are no routers within an ad hoc network.

A host, when booted on an IPv6 link, first creates a *link-local* address by taking the architecturally defined prefix in Neighbor Discovery **FE80**, and appending an *End User Identifier* (EUI), determined by the host, to that prefix. This link-local address is then verified on the link that it is not duplicated with other link-local addresses on that host's link. This host communication is performed using link IPv6 multicast packets, to avoid duplicate link-local addresses, which are not permitted on an IPv6 Link.

The host then uses the link-local address to send on the IPv6 link *Neighbor Solicitations,* and all other hosts on that link see those multicast solicitations, and then return *Neighbor Advertisements* to the host. After this communications process, all nodes on the IPv6 link can now communicate, and communication was accomplished without the use of servers or routers in a stateless manner.

The host also listens for *Router Advertisements* on the IPv6 link (or sends *Router Solicitations*), which provide address prefixes, link configuration parameters, and information as to whether or not to use a stateless or stateful method for address assignment, and additional network configuration parameters using the *Dynamic Host Configuration Protocol for IPv6* (DHCPv6)[1].

If the host is instructed to use the stateless method for address configuration, then it can use the router prefixes announced to form IPv6 addresses from those prefixes by appending the EUI determined from the link-local address to that prefix to create an IPv6 Address. IPv6 supports multiple address types within the address architecture[2,3]. If the host is instructed to use the stateful method for address configuration, then DHCPv6 can be used to configure additional hosts' addresses.

Users will not see these IPv6 stateless advantages for network communications, but they will exist behind the wall of the user to provide a new and improved set of mechanisms for Node Discovery and Address Autoconfiguration far more robust and efficient than using the current IP Version 4 (IPv4) protocol. The IPv6 Stateless Architecture for Node Discovery permits a new model for node communications on links.

The Mobile IPv6 Technology Value Proposition

Mobile IPv6 offers many improvements over Mobile IPv4. Mobile IP as a technology permits users to remain connected across wireline (for example, Ethernet, xDSL) and wireless (for example, 802.11, cellular, satellite) networks, while roaming between networks. This permits users to stay connected while on the way to the airport from home, rather than shutting down their personal digital assistant (PDA)/laptop at home, and reconnecting at the WiFi location at the airport.

Figure 1: Route Optimization with Built-In Security

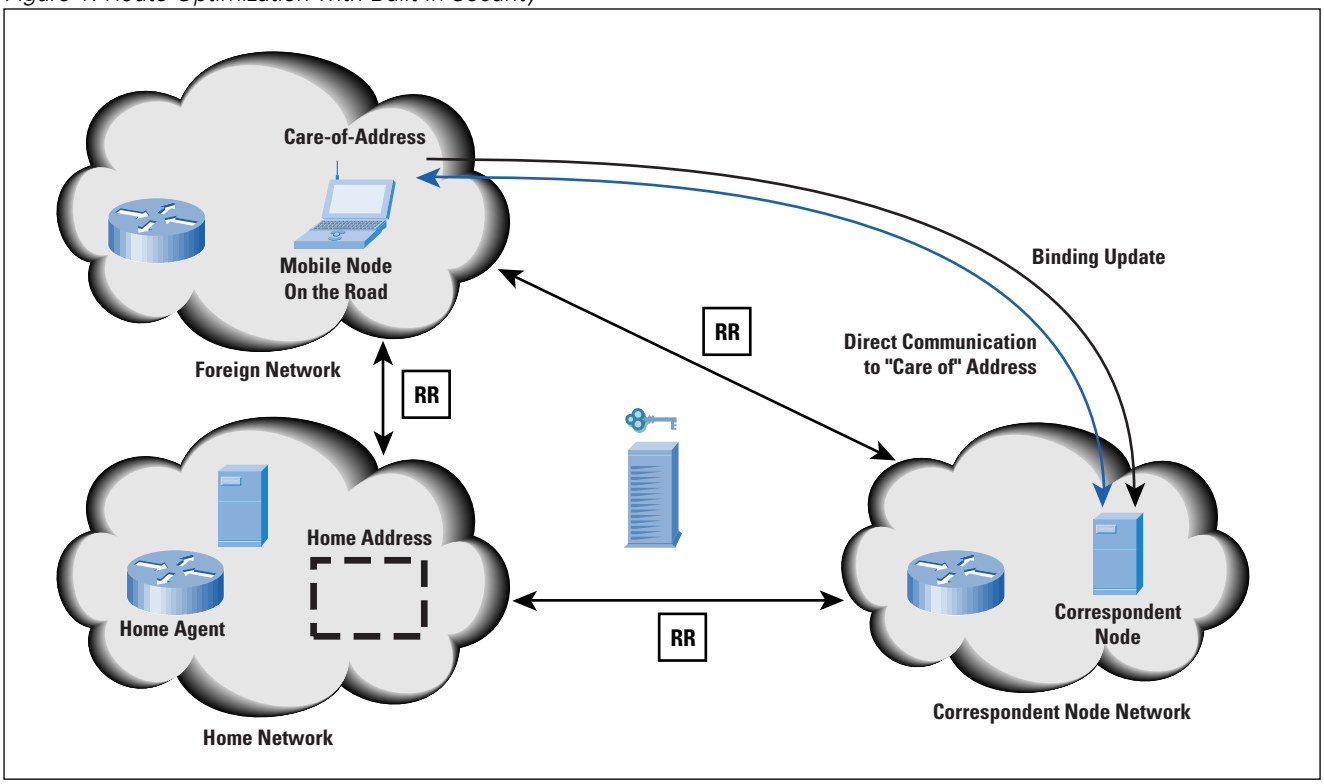


Figure 1 depicts the multiple phases of a mobile IPv6 connection. On the home network, a mobile node receives its home address as any IPv6 node. The mobile node registers that address with the *Home Agent,* which is a router that keeps the location information for the mobile node when it moves to a foreign network, stores the mobile-node *care-of address* when the mobile node is away from home, and performs other functions on behalf of the mobile node when it is away from home. A peer node that the mobile node communicates with is defined as the *Correspondent Node* (which may be stationary or mobile).

Security between the mobile node and home agent can be accomplished using the *IP Security Protocol* (IPSec) architecture. This permits secure communications between the mobile node and the home agent. When a correspondent node receives a packet from a mobile node, it first checks its binding caches to see if it has a cache of the mobile-node care-of address, and if it does not, the correspondent node sends the packet to the mobile-node home address. The home agent receives all packets sent to the mobile node when it is away from home and then tunnels the packets to the mobile-node care-of address.

To permit a mobile node and correspondent node to communicate directly, without going through a home agent, requires the use of *Mobile IPv6 Route Optimization.* First the connection to the correspondent node needs to be secure from the home agent and directly from the mobile node. In the figure, that is done using a procedure defined as *Return Routability* (RR) within the Mobile IPv6 protocol. The network path between the mobile node and correspondent node is secured through the RR procedure.

Mobile IPv6 uses the extensibility of the IPv6 protocol defining new Neighbor Discovery messages and types, *Routing Header*, and the use of the *Destination Option* in an IPv6 packet, which does not exist in IPv4. Discussion of those extensions is beyond the scope of this article, and is left as an exercise for readers to read the actual Mobile IPv6 specification.

Mobile IPv6 has core technical operational advantages over Mobile IPv4, as follows:

- There is no need to deploy special routers as "foreign agents," as in Mobile IPv4. Mobile IPv6 operates in any location without any special support required from the local router.
- Support for route optimization is a fundamental part of the protocol, rather than a set of nonstandard extensions.
- Mobile IPv6 route optimizations can operate securely even without prearranged security associations. It is expected that the route optimizations can be deployed on a global scale among all mobile-node correspondent nodes.
- Support is also integrated into Mobile IPv6 for allowing route optimizations to coexist with routers that perform ingress filtering.
- The IPv6 *Neighbor Unreachability Detection* assures symmetric reachability between the mobile node and its default router in the current location.
- Most packets sent to a mobile node away from home in Mobile IPv6 are sent using an IPv6 routing header rather than IP encapsulation, reducing the amount of resulting overhead compared to Mobile IPv4.
- Mobile IPv6 is decoupled from any particular link layer because it uses IPv6 Neighbor Discovery instead of IPv4 *Address Resolution Protocol* (ARP). This also improves the robustness of the protocol.
- The use of IPv6 encapsulation (and the routing header) removes the need in Mobile IPv6 to manage tunnel soft state.
- The dynamic home-agent address discovery mechanism in Mobile IPv6 returns a single reply to the mobile node. The directed broadcast used in IPv4 returns separate replies from each home agent.

Summary

This article has presented three of the key technology advantages of IPv6 behind the wall. There are others, but they are technically too complex to define in a short article, but rather the subject of IPv6 implementation white papers. The IPv6 architecture extends the potential for the Next-Generation Internet to support rapid renumbering of networks, Quality of Service, extensions for ad hoc networks, and the hope of extending the Internet beyond the capabilities and functions today with IPv4. Most important is that IPv6 enhancements will be developed without using "band aids," as is currently being done with today's IPv4 architecture. The author of this article would like to thank Tony Hain and Patrick Grossetete from Cisco Systems for their review.

For Further Reading

[1] R. Droms, Ed., J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315, July 2003.

[2] R. Hinden, S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture," RFC 3513, April 2003.

[3] R. Hinden, S. Deering, E. Nordmark, "IPv6 Global Unicast Address Format," RFC 3587, August 2003.

Additional information regarding IPv6 can be found at the International IPv6 Forum Web site **www.ipv6forum.com** and the North American IPv6 Task Force Web site **www.nav6tf.org**. Specifically, readers can view the IPv6 Forum basic value proposition at:

http://www.nav6tf.org/summit_slides/IPv6_Value_Proposition_June_2003final.ppt

JIM BOUND works at Hewlett Packard Corporation as an HP Fellow and is a Network Technical Director within the Enterprise UNIX (HP-UX) Division's Network and Security Lab Engineering Group. Jim was a member of the Internet Protocol Next Generation (IPng) Directorate within the IETF, which selected IPv6, among several proposals, to become the basis of the IETF's work on an IPng in 1994. Jim has been a key designer and implementor of IPv6, and contributor and coauthor of IPv6 specifications. Jim founded an ad-hoc IPv6 deployment group working with implementors across the Internet in 1998, which became the IPv6 Forum, where Jim is now Chair of the IPv6 Forum Technical Directorate and Member of the Board of Directors. Jim is also Chair of the North American IPv6 Task Force. Jim is a pioneer member of the Internet Society, and member of the Institute of Electrical and Electronics Engineers (IEEE). In July 2001, Jim received the IPv6 Forum Internet IPv6 Pioneer Award as the IPv6 Forum's "Lead Plumber." Jim has been working in the field of networking as engineer and architect since 1978, and is a subject matter expert to government and industry, for IPv6 and network-centric technology. E-mail: **jim.bound@hp.com**

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. The journal carries tutorial articles ("What is…?"), as well as implementation/operation articles ("How to…"). It provides readers with technology and standardization updates for all levels of the protocol stack and serves as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

- Access and infrastructure technologies such as: ISDN, Gigabit Ethernet, SONET, ATM, xDSL, cable fiber optics, satellite, wireless, and dial systems
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, trouble-shooting, and mapping
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, network computing, and Quality of Service
- Application and end-user issues such as: e-mail, Web authoring, server technologies and systems, electronic commerce, and application management
- Legal, policy, and regulatory topics such as: copyright, content control, content liability, settlement charges, "modem tax," and trademark disputes in the context of internetworking

In addition to feature-length articles, IPJ will contain standardization updates, overviews of leading and bleeding-edge technologies, book reviews, announcements, opinion columns, and letters to the Editor.

Cisco will pay a stipend of US$1000 for published, feature-length articles. Author guidelines are available from Ole Jacobsen, the Editor and Publisher of IPJ, reachable via e-mail at **ole@cisco.com**

Fragments

Peter T. Kirstein Receives Postel Award

Peter Kirstein is this year's recipient of the prestigious *Jonathan B. Postel Service Award*. A founding member of the Internet Society, Professor Kirstein is one of the pioneers of the Internet and was directly involved with its development and evolution. He was awarded the Postel Service Award in recognition of his foresight, persistence and innovation in navigating international technical and political complexities, and thus enabling the global propagation of the Internet. The Postel Award was presented on July 16, during the 57th meeting of the *Internet Engineering Task Force* (IETF) in Vienna, Austria.

"The Internet Society is pleased to recognize Peter's significant contribution to the development of the Internet by awarding him this year's Postel Award," said Internet Society President/CEO Lynn St. Amour. "His commitment to the evolution and growth of the Internet, particularly during the 1970s, made possible the global infrastructure we have today. And, his efforts continue, most recently working in the Southern Caucasus and Central Asia regions." Steve Crocker, noted Internet authority and chair of this year's Postel award committee, commented on Kirstein's foresight in laying the groundwork for the Internet's global scope. "Peter Kirstein saw that the future of networking lay in international cooperation and interconnection, and deftly organized the steps to make it happen. He used both technical and personal skills and enabled many others to do magnificent work."

In 1973, Kirstein established one of the first two international nodes of the ARPANET, playing a very active part in the ensuing SATNET activity, which covered five countries. His group continued to provide the principal Internet link between the UK and the US throughout the 1980s, during which time he was responsible for both the **.UK** and **.INT** domains. He continues to collaborate in US *Defense Advanced Research Agency* (DARPA) programs. He has led six European projects in computers and communications funded by the European Commission, and participated in twelve more. Currently, he is leading the *Silk Project,* which is providing satellite-based Internet access to the Newly Independent States in the Southern Caucasus and Central Asia. In June, he was awarded a *Commander, Order of the British Empire,* for his services to Internetworking research.

He has chaired the International Collaboration Board, which currently involves six NATO countries, since 1983, and served on the Networking Panel of the *NATO Science Committee* (serving as chair in 2001). He has been on Advisory Committees for the *Australian Research Council,* the *Canadian Department of Communications,* the German GMD, and the Indian *Education and Research Network* (ERNET) Project. Kirstein obtained his undergraduate degree in Mathematics and Engineering from Gonville and Caius College, Cambridge University, his PhD in Electrical Engineering from Stanford University, and was awarded a DSc in Engineering from the University of London.

Kirstein expressed his appreciation for the award and respect for Jon Postel's work, explaining, "Postel's efforts to ensure the successful development and deployment of the Internet was an inspiration to us all. His stewardship of the RFC series was essential to the successful development of the Internet. His conscientious and painstaking operation of the Domain Name System and the Internet Assigned Numbers Authority were indispensable to the international growth of the system. I am particularly pleased to be recipient of an award in his name, and feel greatly honored to be considered worthy of having my activities linked with his memorial."

The Jonathan B. Postel Service Award was established by the Internet Society to honor those who have made outstanding contributions in service to the data communications community. The award is focused on sustained and substantial technical contributions, service to the community, and leadership. With respect to leadership, the nominating committee places particular emphasis on candidates who have supported and enabled others in addition to their own specific actions.

The award is named after Dr. Jonathan B. Postel, who embodied all of these qualities during his extraordinary stewardship over the course of a thirty-year career in networking. He served as the editor of the RFC series of notes from its inception in 1969, until 1998. He also served as the ARPANET "numbers Czar" and the Internet Assigned Numbers Authority over the same period of time. He was a founding member of the *Internet Architecture* (nee *Activities*) *Board* (IAB) and the first individual member of the Internet Society, where he also served as a trustee.

Previous recipients of the Postel Award include Jon himself (posthumously and accepted by his mother), Scott Bradner, Daniel Karrenberg and Stephen Wolff. The award consists of an engraved crystal globe and $20,000.

The *Internet Society* (ISOC) (**`www.isoc.org`**) is a not-for-profit membership organization founded in 1991 to provide leadership in Internet related standards, education, and policy. With offices in Washington, DC, and Geneva, Switzerland, it is dedicated to ensuring the open development, evolution and use of the Internet for the benefit of people throughout the world. ISOC is the organizational home of the IETF, the IAB, the *Internet Engineering Steering Group* (IESG) and other Internet-related bodies who together play a critical role in ensuring that the Internet develops in a stable and open manner. For over 12 years ISOC has run international network training programs for developing countries and these have played a vital role in setting up the Internet connections and networks in virtually every country connecting to the Internet during this time.

Deployment of Internationalized Domain Names

The *Internet Corporation for Assigned Names and Numbers* (ICANN) recently announced the commencement of global deployment of *Internationalized Domain Names* (IDNs)[2,3,4], which will allow use on the Internet of domain names in languages used in all parts of the world.

In October 2002, the IESG approved the publication of a standardized way of integrating IDNs into the Internet's *Domain Name System* (DNS). After the proposed technical standard was published in March 2003, the ICANN Board endorsed an approach for implementation of the technical standard that had been developed cooperatively by ICANN and leading IDN registries.

Following up on the Board's endorsement, ICANN and the leading IDN registries finalized an agreed text of the principles to be followed in IDN registration activities. Those "Guidelines for the Implementation of Internationalized Domain Names"[1] were published. IDN registries adhering to the Guidelines will employ language-specific registration and administration rules that are documented and publicly available. These IDN registries will work collaboratively with each other and with interested stakeholders to develop the language-specific policies, with the objective of achieving consistent approaches to IDN implementation to maintain Internet interoperability for the benefit of DNS users worldwide.

The registries for the **`.cn`** (China), **`.jp`** (Japan), and **`.tw`** (Taiwan) country codes, as well as for the **`.info`** and **`.org`** generic top-level domains, have committed to adhere to the Guidelines. As authorized by the ICANN Board in March, registries seeking to deploy IDNs under their agreements with ICANN will be authorized to do so on the basis of the Guidelines. In addition, the ICANN Board has recommended the Guidelines to other registries, and encourages broad participation by registries, language experts, and others in consultative, collaborative, community-based processes to study and develop appropriate language-specific IDN registration rules and policies.

As the deployment of IDNs proceeds, ICANN and the participating IDN registries have agreed to work together to review Guidelines at regular intervals based on their deployment experience, and to make any necessary adjustments.

[1] **`http://www.icann.org/general/idn-guidelines-20jun03.htm`**

[2] P. Faltstrom, P. Hoffman, A. Costello, "Internationalizing Domain Names in Applications (IDNA)," RFC 3490, March 2003.

[3] P. Hoffman, M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)," RFC 3491, March 2003.

[4] A. Costello "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)," RFC 3492, March 2003.

The Internet Protocol Journal

Ole J. Jacobsen, Editor and Publisher

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

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive, M/S SJ-7/3
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
Cisco Systems, Inc.