The Internet Protocol Journal

March 2017 Volume 20, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.protocoljournal.org

ISSN 1944-1134

From The Editor

The *Regional Internet Registries* (RIRs) form an important part of the administrative ecosystem of the Internet. APNIC and RIPE are already sponsors of this journal, and we are pleased to announce that another RIR, the *Latin America and Caribbean Network Information Centre* (LACNIC), is now a sponsor of IPJ. LACNIC has additionally agreed to translate selective articles and provide article summaries from IPJ in Spanish.

Selective articles from IPJ are also available in Russian through the publication Интернет изнутри (*Internet Inside*), available at:
http://www.ccni.ru/publications/

Our individual donors and organizational sponsors make publication of this journal possible. We are especially thankful for the support of Rabbi Rob Thomas and Lauren Thomas who agreed to match individual donations from October 2016 until April 2017.

The *Border Gateway Protocol* (BGP) is a core component of the Internet routing system. Like most Internet protocols, BGP was developed in the *Internet Engineering Task Force* (IETF). The IETF has been criticized for its slow process, and in many cases for developing protocols without proper input from those who actually run networks, collectively referred to as the "operators." Job Snijders describes how a group of dedicated operators were able to develop specifications for *BGP Large Communities* within the IETF process in record time.

Various aspects of *The Internet of Things* (IoT) have previously been covered in this journal. This time Bob Hinden looks at the problem of securing IoT devices in light of some large-scale attacks that exploited security weaknesses in common devices such as IP cameras.

In our final article, Geoff Huston and Joao Luis Silva Dama discuss *privacy* in the context of the *Domain Name System* (DNS). The *DNS PRIVate Exchange* (DPRIVE) Working Group of the IETF has been working on this topic, considering ways in which the interaction between a DNS client and a DNS resolver can be protected.

Visit our website for subscriptions, back issues, author guidelines, sponsor information, and much more, and send us your feedback via e-mail to **ipj@protocoljournal.org**.

—Ole J. Jacobsen, Editor and Publisher
ole@protocoljournal.org

BGP Large Communities

by Job Snijders, NTT Communications

If the *Domain Name System* (DNS) is the phonebook of the Internet, used to translate names to numeric addresses, then the *Border Gateway Protocol* (BGP)[1] is the map of the Internet: how to reach those addresses. This article focuses on a minute, but oh-so-critical aspect of BGP operations called *BGP Communities*. We will cover what BGP Communities are, why "done" is better than "perfect," and how disaster was avoided.

A BGP crash course: The Internet is an assemblage of independent networks. The technical term for such a network is an *Autonomous System* (AS), and each is assigned a globally unique identifier called the *Autonomous System Number* (ASN). All these networks exchange routing information with other networks using BGP, a path vector protocol. This exchange of routing information is composed of announcements such as "this specific set of addresses can be reached through my network." A set of addresses is called a *route*. Routes are often decorated with meta-information known as BGP Communities. These BGP Communities can be considered marker colors of sorts. Such marker colors are used as an additional input in the route evaluation process, which is a function of a network routing policy (Figure 1).

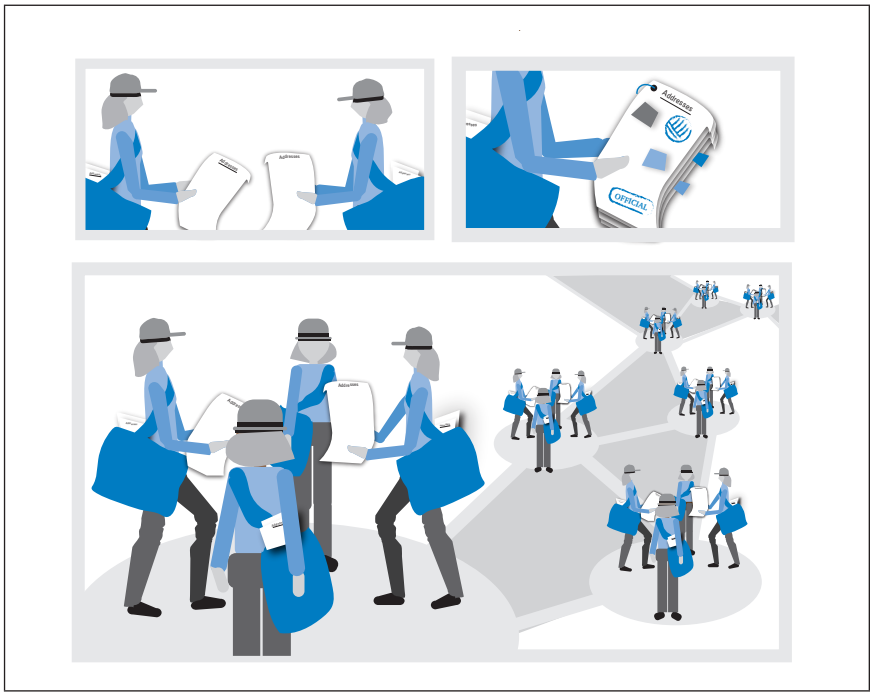


Figure 1: The Stamps on the List of Addresses Symbolize the Function of BGP Communities

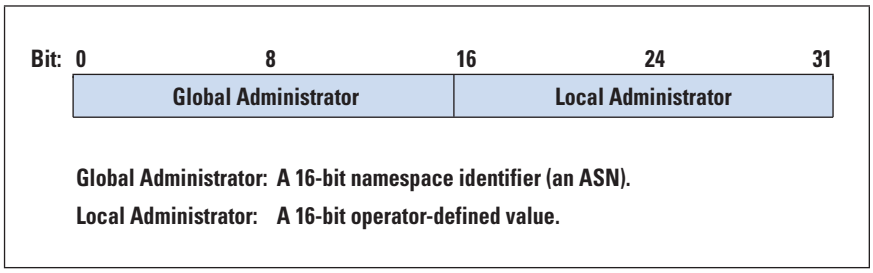
Over the last 20 years, BGP Communities have become the tool of choice to facilitate operations in all BGP networks. BGP Communities are used, for instance, to group routes together and ensure that only a specific group of routes is announced at a given interconnection point with another network.

However, along with the advent of a new type of network identifier in 2007, 32-bit ASNs (previously ASNs were 16 bits) came an operational problem: by embracing *32-bit ASNs*, we had suddenly outgrown classic BGP Communities.

How Can BGP Communities Be Too Small?

Let's first explore what BGP Communities actually mean for operations. BGP Communities are defined in RFC 1997[2]. Each BGP Community is a fixed-width 32-bit entity; you can attach multiple BGP Communities to a route. The convention is that the first 16 bits are the ASN in which the last 16 bits have a meaning. For human consumption, a BGP Community is usually represented as two 16-bit values separated by a colon (Figure 2).

Figure 2: A Classic BGP Community



An example BGP Community and its application would be `2914:664`; the first part, `2914`, is NTT Communications' ASN. The second part (called the *Local Administrator*), which carries the value `664`, is defined by NTT and has meaning only within NTT's network. According to NTT's documentation[3], the value `664` within the `2914` namespace means "only blackhole outside the country the announcement originated." BGP Communities are the lingua franca of inter-domain routing; however, (oddly enough) its vocabulary is exchanged through out-of-band means such as published documentation. Thousands of networks have defined their own routing functions and associated BGP Communities.

In the early 2000s work began to extend the BGP protocol so that it could accommodate the ever-growing Internet. The exhaustion of the 16-bit ASN pool we are facing right now was already anticipated then. Through RFC 4893[4] the range of possible ASNs was extended from 65,535 to 4,294,967,295 ($2^{32} - 1$). The observant reader will have noted the friction between the previously described application of a BGP Community and the existence of 32-bit ASNs: you simply cannot fit a 32-bit value in a 16-bit field!

Thus, operators of networks with a 32-bit ASN have been forced to work around this problem. Operators have used kludges ranging from using private 16-bit ASNs in the "ASN" field (those first 16 bits), to the ultimate rejection of the concept of 32-bit ASNs: returning the assigned 32-bit ASN to its respective *Regional Internet Registry* (RIR) and requesting a fresh 16-bit ASN. However, the *Internet Assigned Numbers Authority* (IANA) has been depleted of its supply of 16-bit ASNs. The RIRs are making impossible searches for 16-bit ASNs.

Rumor has it that some RIRs were considering reclamation strategies to increase their pool of 16-bit ASNs. A dire situation!

The Road to Perfection Is Always Under Construction

Surely the *Internet Engineering Task Force* (IETF) thought of this situation when updating the BGP-4 specification for 32-bit ASNs? Absolutely! But not in a way that would match operational practices. RFC 4893[4] deferred the issue of BGP Communities as follows: "... the high-order two-octets of the community attribute [...] encode the Autonomous System number." Quite clearly this would not work for BGP speakers that use 4-octets Autonomous System Numbers. Such BGP speakers should use the four-octet AS-Specific *Extended Communities* instead (see Figure 3).

Yes, Extended Communities, as defined in RFC 4360[5], are bigger than regular BGP Communities: they contain a *Type* and *Subtype* field followed by 48 bits of data. However, the 48-bit length of the Extended Community value precludes the common operational practice of having the ability to encode ASNs in both the Global Administrator and the Local Administrator subfields. You can either encode a 16-bit value in the first part and a 32-bit value in the second part (Figure 3), or the other way around, but not a 32-bit value in both fields (Figure 4).

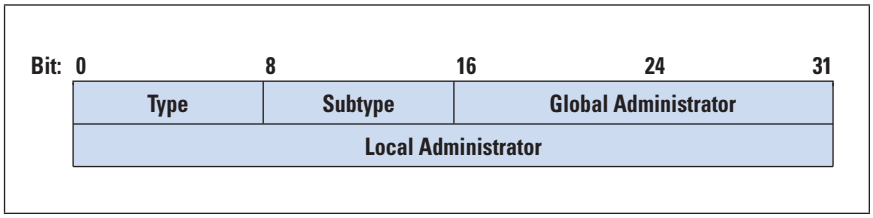


Figure 3: A BGP Extended Community Flavor #1

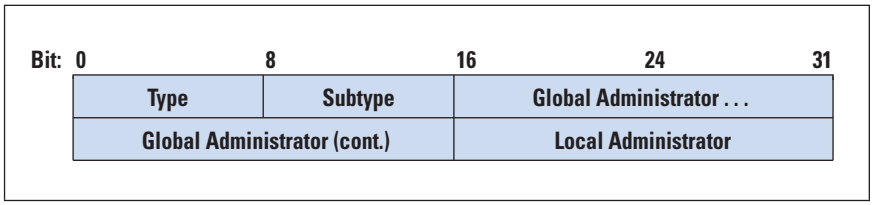


Figure 4: A BGP Extended Community Flavor #2

The Type and Subtype fields define the kind of Extended Community. The Value field can be split into either a two-octet *Global Administrator* subfield and a four-octet *Local Administrator* subfield, or a four-octet Global Administrator and a two-octet Local Administrator.

Even if your network does not have a 32-bit ASN, you might have to interact with 32-bit ASN networks. If we go back to the example of NTT's routing policy[3], one of the traffic engineering features is exposed through BGP Communities `65501:nnn`, where nnn is to be replaced with the Peer ASN and `65501` means "prepend AS 2914 once on outbound." Ideally this works for 32-bit ASNs too!

The previous example highlights two issues: apparently AS 2914 ran out of BGP Community space and resorted to using a Private ASN in the first 16 bits: `65501`; this situation is considered a form of namespace pollution. And secondly, had 2914 instead been a 32-bit ASN (such as AS 199036), you would not be able to encode a 32-bit ASN in the second field, even with Extended Communities. In other words, you cannot fit 64 bits worth of information into 48 bits of room. You can find many examples[6] of networks offering traffic engineering features through the verbatim reference of a target Peer ASN in the BGP Community itself. Extended Communities have excellent use cases, but simply put, they aren't big enough for Internet routing operations.

So, if both Communities and Extended Communities weren't suitable, a third reimagination of the Communities *concept* was needed. Since these technologies tend to last for decades, and the previous two iterations had proven not to be usable if the technology changes, the bar was set pretty high.

Notable efforts in this problem space include *Flexible Communities* (started in 2003) and *Wide Communities* (started in 2010). These efforts have highlighted a disconnect between the IETF and operator communities. Not only were these two efforts moving forward on vastly different timescales than the operational community required (keep the impending doom of 16-bit ASN exhaustion in mind), but both efforts presented a tendency towards *feature creep*. The limitless extensibility was both a virtue and a curse: every possible use case would be consumed effortlessly in the specification, so the schedule overran, the specification complexity increased, and as a result no actual implementations had been produced. "A bird in the hand is worth two in the bush."

Another anti-pattern was at work in the IETF *Inter-Domain Routing* (IDR) Working Group. The anti-pattern is commonly called *Cookie Licking*. Cookie Licking is a reference to the metaphorical situation where someone takes a cookie, licks it, puts it back on the tray, and does not eat it! In volunteer communities this phenomenon can be noticed when a certain topic is discussed and someone says "I've already got this covered in my Internet-Draft." This statement might defer others from working on the topic, since you could easily assume the problem will be taken care of. But with the 32-bit ASN specification approaching the respectable age of 10 years, the pool of 16-bit ASNs running out, and no commonly acceptable successor available for BGP Communities, it became increasingly clear that a catastrophe was imminent. The issue had become a matter of extreme urgency: if we didn't start turning the tanker right now, it would crash into a wall two years down the road.

Origin Story: BGP Large Communities

In March 2016 in Sweden, Ignas Bagdonas (Equinix) presented his perspective on "some form of larger BGP communities." In this presentation he iterated over challenges that 32-bit ASNs network operators face and provided an up-to-date overview of current and past attempts to mitigate those issues. After this presentation, I approached Bagdonas and proposed to team with him and jointly drive this effort forward.

In April 2016, I flew to the United States, where my friend Jared Mauch introduced me to Jakob Heitz (Cisco) over lunch. Heitz was intrigued by the effort and committed to providing running code for some form of "a simple, bigger BGP Community." This agreement meant the first router vendor got on board, even if we didn't know what the results would look like!

In May 2016 at the RIPE 72 meeting, Bagdonas presented again in the *Routing Working Group*. In the Q&A session, Ruediger Volk (Deutsche Telekom) made one of the most formative comments on the "larger communities" effort. Volk said: "The discussions in IETF about extending this [communities] have been around for many years, and no progress has been made. Any proposal that has an extended functionality comes with the problem that discussion of the additional functionality does not have a proof of termination."

In other words, the IETF suffers from *bikeshedding*[7]. Only something that was purposefully specified to be as narrow and as simple as possible would meet the network operator community's immediate needs. Volk went on to say that something similar to the opaque approach of RFC 1997[2] should be done, where the first bits indicate "Who owns the namespace" followed by some extra bits for the actual routing policy work.

It was this specific argument about namespace that led to defining Large BGP Communities as a 96-bit entity: All operators whether they have a 16-bit or a 32-bit ASN, would have 64 bits (8 bytes) of room to signal information or trigger actions in their network. With 64 bits available to the network operator, there even is enough to target a 32-bit ASN and still have space for an action such as "prepend" or "do not export."

These conversations in the operational community led directly to Large Communities (by design) not being extendable. Extensibility comes at a cost. Also, knowing that the amount of noise generated by an idea is inversely proportional to the complexity of the idea, the IETF was asked to consider the simplicity of the Large Community a virtue, not a disadvantage. Of course, that request was ignored: over the span of just four months, almost 5% of all emails ever sent through the IDR mailing list in the last 20 years addressed the topic of Large Communities.

Aside from the fierce debate on the actual technology itself, the IANA *Early Allocation Procedure* uncovered very serious deployment issues. Initially Large Communities were assigned *BGP Path Attribute* value 30 by the IANA. Immediately following this allocation, BGP Beacons with the Large Community attached were brought online[8] to test for problems of any kind. We expected the beacon prefix at least to propagate to all corners of the Internet, and optionally with its *Transitive Large BGP Communities* attribute still attached and intact. The beacon was launched from my home, but unfortunately the assigned codepoint did not pass the "Family Acceptance Factor." My significant other noticed how certain websites that were reachable the day before could be reached no more. Collective sleuthing of the NLNOG[9] operators brought to light that certain Huawei software was using BGP Path Attribute value 30 for something entirely different than Large Communities, and as such legitimate Large Communities were considered invalid and treated such prefixes with a withdraw. After this knowledge became public, Cisco and Juniper also emerged and a grand total of six squatted BGP Path Attribute values were uncovered[10]. Squatting in this context means that the code point is used for a different purpose than the one designated by the IANA. In the end BGP Large Communities was assigned a new value: `32` — a very befitting number for the problem it solves.

To end this origin story: Ruediger Volk provided the effort with a challenge: "If we go really fast-track, we'll be there in 5 years' time, not before." We all know that nothing motivates as much as a solid race against the clock: In exactly 6 months the Large Communities team produced 18 versions of the technical specification[11], patched 2 packet analyzers, developed 7 implementations[12], and obtained approval from the *Internet Engineering Steering Group* (IESG) to publish the document as an IETF Standards Track RFC.

So What Is This "BGP Large Community"?

An example of a Large Community is `2914:65400:38016`. Each BGP Large Community value is encoded as a 96-bit quantity: three unsigned 32-bit integers, separated by a colon. The `2914` part is called the *Global Administrator,* and the second and third fields (`65400` and `38016`, respectively) are called *Local Data Part 1* and *Local Data Part 2* (Figure 5).

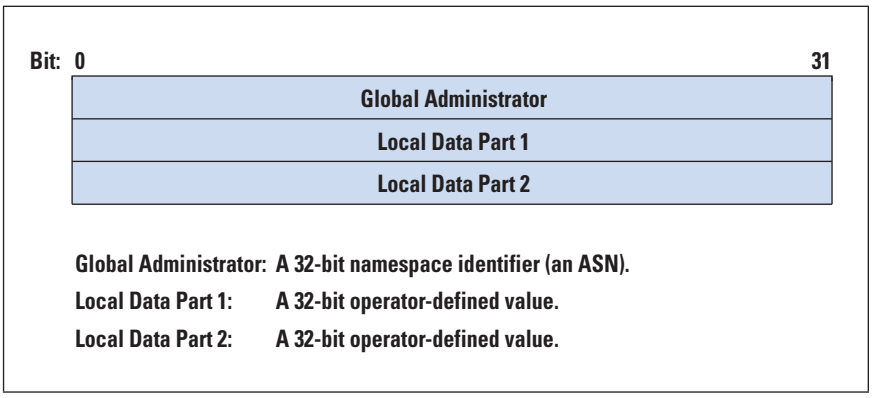


Figure 5: A BGP Large Community

The Global Administrator is a 32-bit namespace identifier that allows different Autonomous Systems to define Large Communities without collision. The recommendation is to put your RIR-assigned ASN in the Global Administrator field. Using a 32-bit field allows full parity and fairness between 16-bit and 32-bit ASNs. Everyone can use their own ASN in Large Communities. The "Local Data Parts" are to be interpreted as defined by the owner of the ASN.

Special care was taken to define a canonical representation for Large Communities. Especially in our international community where we face communication challenges because of language barriers, it is important that Large Communities are easy to remember and easy to communicate by phone or email.

Use of BGP Large Communities

A design pattern promoted by the Internet Draft **`draft-ietf-grow-large-communities-usage`**[13] specifies a **`ASN:Function:Parameter`** pattern to fill the three Large Community fields.

In existing deployments of Communities RFC 1997[2] and preliminary deployments of Large Communities, two categories of Communities exist: *Informational Communities* and *Action Communities*.

Informational Communities serve as markers regarding, for instance, the origin of the route announcement, the relation with the *External Border Gateway Protocol* (EBGP) neighbor, or the intended propagation audience. Informational Communities also assist in network operations such as debugging. The Global Administrator field is set to the ASN that marks the routes with the Informational Communities. As an example: on a route that AS 64497 announces to AS 64498, AS 64497 might add Large BGP Community **`64497:100:31`** to signal to AS 64498 that the route was learned in the Netherlands. In this instance, the **`100`** value in *Local Data Part 1* is an indicator for the function "in which country a route originated" and the value **`31`**, as parameter, symbolises the Netherlands. In general, the intended audience of Informational Communities is downstream networks, but any Autonomous System could benefit from receiving these communities.

Action Communities are attached to routes to request nondefault behaviour in an Autonomous System. For instance, Action Communities are used to change the propagation characteristics of the route, or BGP Path attributes such as *LOCAL_PREFERENCE, AS_PATH*, and so forth. The Global Administrator field is set to the ASN value of the AS that has defined the meaning of the remaining fields and is expected to perform the action upon receiving the route. For instance, if AS 64499 wants to request AS 64497 to lower the *LOCAL_PREFERENCE* to **`50`** (below the default of **`100`**), AS 64499 could tag the route with **`64497:20:50`**. In general the intended audience of Action Communities is an upstream provider.

A real-life example of the application of Large BGP Communities can be found at the Route Servers operated by the *Internet Neutral Exchange Association* (INEX), Ireland. INEX[14] extended its Route Server routing policy to support control over which Peer ASN receives what routing information through a trivial suppress/unsuppress mechanism. For instance, Large Community `43760:1:peer-as` means "Announce prefix to a peer-as," whereas `43760:0:peer-as` means "Prevent announcement of a prefix to the ASN filled in as peer-as." With BGP Communities such signaling was exclusively applicable to 16-bit ASNs.

Implementations

An IETF attitude from the past is "rough consensus and running code," and especially for a technology like Large Communities (designed to be used in an inter-domain context), the more implementations, the more stable and commonly accepted the standard is. At the moment of writing there are nine confirmed implementations in various stages of general availability.

It is clear that the open source projects have taken a lead in implementing Large Communities. I attribute this situation in part to the fact that any contributor can dedicate time to create a patch. Secondly, the size of an ecosystem and the expected self-reliance (if any) of the user base affect the size of the effort. After all, most open source projects won't need, or won't have to train pre- and post-sales staff and technical assistance centers, update volumes of internal and external documentation, and, last but not least, figure out how to port the feature into the many concurrently supported releases.

These open source projects benefited from an open regression testing suite: *The Large Communities Playground*[15]. This open source cross-vendor effort brought easy specification compliance, and functional testing to any Open Source Software project runnable inside Docker. The availability of this tool prevented duplicate efforts in establishing the appropriate environments, and ensured consistent interoperability of the implementations.

Considering the information available to me right now: *ExaBGP, GoBGP, OpenBGPD, rtbrick,* and *pmacct* are shipping Large Communities in their stable releases. *Quagga* and *frr* ship today or will do so shortly. *Arista EOS, Cisco IOS XR,* Juniper's *Junos* OS, and Nokia *SR* OS are expected to publish software in the second half of 2017. Ancillary parts of the ecosystem such as *tcpdump, Wireshark, pbgpp, zebra-dump-parser, bgpdump,* and *mrtparse* have also been updated. Support for Large Communities in just the BGP Speakers simply wouldn't be enough: to be a viable alternative to BGP Communities, every element in the ecosystem has to be updated, from packet analysers to research tools to statistics backends.

The future looks bright…2018 will be the year of Large Communities.

Conclusion

This challenging crusade through the IETF process has changed me from being an outside operator to a participant in the standardisation community. As much as Large Communities are a part of me, I am now a part of the IETF. Any criticism against the IETF institution displayed in this article applies equally to me as it does to others. We need to be vigilant to prevent another occurrence where operational reality and standardisation efforts grow apart so far that the only recourse left is to round up all the operators and show up with pitchforks and torches. In the end, it's just the results that matter; anything else is unimportant. Large Communities are here now, and will be so for the next decades.

Large Communities closed the final feature-parity gap between 16-bit and 32-bit ASNs. Now 32-bit ASNs are first-class citizens too.

Acknowledgements

I would like to thank Arjen Zonneveld, Saku Ytti, Russ White, Ruediger Volk, Teun Vink, Gunter van de Velde, Jeff Tantsura, Sander Steffann, Richard Steenbergen, Wesley Steehouwer, Adam Simpson, Rob Shakir, Shyam Sethuram, Julian Seifert, Mark Schouten, Tom Scholl, Martijn Schmidt, Alvaro Retana, Kay Rechthien, Gaurab Raj Upadhaya, Joe Provo, Stefan Plug, Tom Petch, Jussi Peltola, Keyur Patel, Barry O'Donovan, Arnold Nipper, Christopher Morrow, Remco van Mook, Martin Millnert, Jared Mauch, Marco Marzetti, Ben Maddison, Joel M. Halpern, Duncan Lockwood, Acee Lindem, Kristian Larsson, Warren Kumari, Thomas King, Grzegorz Janoszka, Geoff Huston, Paul Hoogsteder, Nick Hilliard, Wim Henderickx, John Heasley, Markus Hauschild, Richard Hartmann, Greg Hankins, Jeffrey Haas, David Freedman, Bill Fenner, David Farmer, Matt Griswold, Bertrand Duvivier, Linda Dunbar, Brad Dreisbach, Gert Doering, Peter van Dijk, Brian Dickson, Ian Dickinson, Marco Davids, Adam Davenport, Tom Daly, Nabeel Cocker, Mach Chen, Adam Chappell, Pier Carlo Chiodi, Randy Bush, Jay Borkenhagen, Theodore Baschak, Niels Bakker, Jan Baggen, Ignas Bagdonas, Alexander Azimov, Eduardo Ascenco Reis, Mikael Abrahamsson, John Scudder and Susan Hares for their support in developing BGP Large Communities. Special thanks to Corinne Pritchard for the imagery.

References

[1] Rekhter, Yakov, Hares, Susan, and Li, Tony, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, January 2006.

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

[3] "NTT Routing Policies," `https://www.us.ntt.net/support/policy/routing.cfm`

[4] Chen, Enke and Vohra, Quaizar, "BGP Support for Four-octet AS Number Space," RFC 4893, May 2007.

[5] Sangli, Srihari R. and Rekhter Yakov, "BGP Extended Communities Attribute," RFC 4360, February 2006.

[6] "BGP Communities Guide," **https://onestep.net/communities/**

[7] "Why Should I Care What Color the Bikeshed Is?" **http://bikeshed.com/**

[8] Snijders, J., "Large BGP Communities Beacon," October 2016. **http://largebgpcommunities.net/2016/beacon/**

[9] Netherlands Network Operator Group, **https://nlnog.net/**

[10] Snijders, J., "Deprecation of BGP Path Attribute Values 30, 31, 129, 241, 242, and 243," RFC 8093, February 2017.

[11] Heitz J., Snijders, J., Patel, K., Bagdonas I., and Hilliard H., "BGP Large Communities," RFC 8092, February 2017.

[12] "BGP Large Communities Implementations," **http://largebgpcommunities.net/implementations/**

[13] Snijders, J. and Schmidt, M., "Usage of BGP Large Communities," Internet Draft, Work In Progress, December 2016. **https://tools.ietf.org/html/draft-ietf-grow-large-communities-usage**

[14] "INEX Deploys Large BGP Communities In Production," November 2016, **http://largebgpcommunities.net/2016/inex-first-in-production/**

[15] Chiodi, P. C. and Snijders, J., "The BGP Large Communities Playground," January 2017, **https://labs.ripe.net/Members/pier_carlo_chiodi/the-large-bgp-communities-playground**

JOB SNIJDERS is IP Development Engineer at NTT Communications, where he analyzes and architects NTT's global IP network for future growth. He has been actively involved in the Internet community in an operational capacity, as a frequent presenter at network operator events, and in numerous community projects for over 10 years. Job is founder of the NLNOG Foundation and vice president of PeeringDB. Job's special interests are routing policy, routing security, and large-scale BGP deployments. He maintains several tools such as *irrtree* and *irrexplorer,* and is active in the IETF, where he has coauthored or contributed to RFCs and Internet-Drafts. E-mail: **job@ntt.net**, Twitter: *@JobSnijders*

The Internet of Insecure Things

by Bob Hinden, Check Point Software

We have a problem. As we have learned recently, most *Internet of Things* (IoT) devices are not secure. They have numerous security weaknesses, including default login/passwords and fixed firmware login/passwords (not easily changeable). In addition, the devices do not get regular software updates to fix security problems, and there is no one to call if problems arise. To make matters worse, these devices exist in large numbers. Gartner says that 6.4 billion IoT devices are deployed now, and forecasts 20.8 billion in 2020. The IoT contains billions of insecure things.

The security problem with these devices is not theoretical; the devices are used today to launch very large-scale *Distributed Denial of Service* (DDoS) attacks. Malware that takes advantage of the security weaknesses of these devices to create botnets exists.

IoT-Based DDoS Attacks

The first large IoT-based attack I am aware of was the DDoS attack on the popular online security publication *KrebsOnSecurity* blog[1]. This attack happened on September 13, 2016. The blog had recently published a series of articles about an organization called *vDOS*, a DDoS-for-hire service. The blog stated that vDOS made approximately $600k in 2 years by knocking sites offline. Two weeks after the articles were published, KrebsOnSecurity was attacked with 620 Gbps of traffic[2]. Akamai was providing pro-bono DDoS protection to KrebsOnSecurity, but it couldn't continue to handle the traffic load[3]. Fortunately, *Google Project Shield*[4] is now protecting the website. Apparently, the source of the attack was IoT devices that included a myriad of technologies composed of IP cameras, *Digital Video Recorders* (DVRs), home routers, and other embedded computers.

The next large IoT-based DDoS attack occurred in early October of 2016. OVH[5] is a large international web-hosting provider. The company was attacked by a botnet comprising more than 145,000 compromised IP cameras and digital video recorders. The attack peaked at 1 Tbps, and fortunately, OVH was able to withstand the attack. This attack was the largest DDoS attack at the time.

A very visible attack occurred on Friday, October 21, 2016. This attack was directed against DYN, a large provider of *Domain Name System* (DNS) services[6]. The attack limited access to many of DYN's customers, including some of the biggest sites on the Internet, like Twitter, Amazon, Tumblr, Reddit, Spotify, and Netflix. For many Internet users, the Internet was down. Table 1 lists the major sites that were affected.

Table 1: Major Sites Affected in the Dyn Attack

Airbnb	Amazon.com	Ancestry.com	The A.V. Club	BBC
Boston Globe	Box	Business Insider	CNN	Comcast
CrunchBase	DirecTV	Elder Scrolls	Electronic Arts	Etsy
EQAO	FiveThirtyEight	Fox News	Guardian	GitHub
Grubhub	HBO	Heroku	HostGator	iHeartRadio
Imgur	Indiegogo	Mashable	NHL	Netflix
NYT	Overstock.com	PayPal	Pinterest	Pixlr
PlayStation	Qualtrics	Quora	Reddit	Roblox
Ruby Lane	RuneScape	SaneBox	Seamless	Second Life
Shopify	Slack	SoundCloud	Squarespace	Spotify
Starbucks	Storify	Swedish Civil Contingencies Agency	Swedish Government	Tumblr
Twilio	Twitter	Verizon	Visa	Vox Media
Walgreens	WSF	Wikia	Wired	Wix.com
WWE	Xbox Live	Yammer	Yelp	Zillow

We are still learning more about this attack, but it appears to be similar to the two earlier attacks. It's been reported that it peaked at 1.2 Tbps, another record. However, unlike the first two attacks, the effects of this one were visible to many Internet users, and it made the news cycle. We don't know for certain the motivations for this attack, but there is speculation that it was due to DYN's support of Brian Krebs, the man behind KrebsOnSecurity. It certainly got everyone's attention.

IoT Devices Involved in DDoS Attacks

The devices used in these attacks include a range of IP cameras, DVRs, and home routers. What these devices share is that they all ship with default login/passwords and can be enabled without changing these defaults. Some even have fixed firmware login/passwords that can't be changed. They are all widely available and fairly low-cost. For example, a search on Amazon for "IP camera" will result in several hundreds of pages of IoT devices for sale, most less than $100 USD. While there clearly has been a lot of innovation to build so many different types and styles of IP cameras, innovation clearly hasn't addressed making them secure.

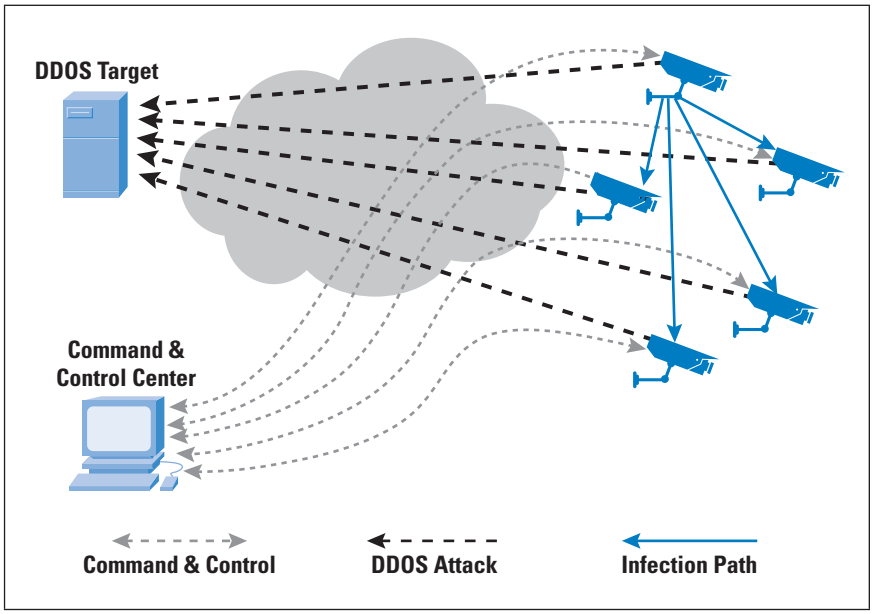
IoT Malware

The malware used in the KrebsOnSecurity attack was released recently. It is called *Mirai*[7]. It works by scanning the Internet for vulnerable devices, looking for systems with factory default usernames and passwords. It installs itself in these IoT devices, turning devices into "bots" used to launch these DDoS attacks. *Bashlight* is another form of IoT botnet malware. It's similar to Mirai, as it infects IoT devices with default usernames and passwords.

According to research from Level3 Communications, Bashlight is responsible for infecting almost a million IoT devices and now competes with other botnets based on Mirai.

Mirai has two phases; in the first phase it infects IoT devices like IP cameras. In the second phase the attack is directed from a *Command and Control System*. These phases are illustrated in Figure 1.

Figure 1: Mirai Attack Mechanism



In the first phase, it scans broad ranges of IP addresses looking for devices that allow it to log in with a set of default login name and password combinations. These default login and password combinations are derived from the documentation of IoT devices. These attacks are very simple, not very sophisticated.

Examples of these login and password combinations follow:

```
admin/admin
root/admin
root/88888
root/root
ubnt/ubnt
```

The problem is that while these IoT devices allow the user to change the default, many users don't, and the setup of these IoT devices does not require it. Thus they are very vulnerable to being taken over by malware like Mirai.

One interesting part of the address scanning is that Mirai has a list of IP address ranges it does not scan. Some of these ranges belong to large companies like General Electric and Hewlett Packard, the U.S. Department of Defense, and the U.S. Postal Service. It's hardly the complete set of addresses for these organizations; I assume the Mirai authors had some reason to avoid these addresses. While the reason is unknown, they may have been trying to avoid detection.

A group named *Malware Must Die*[8] has discovered new variants of IoT malware. This discovery isn't too surprising since the source code for Mirai was released and a lot of malware is a variant of earlier malware. One of these variants is called *Linux/IRCTelnet*[9]. Its attacks are similar to those of Mirai, but it uses *Internet Relay Chat* (IRC) to communicate with compromised Linux-based IoT devices. It also has the capability to attack IoT devices running IPv6.

We Should Be Worried

We should be worried for three fundamental reasons. First, the sheer scale of these attacks is enormous. As shown in Table 2, they are growing:

Table 2: Growth of Attacks

Site	Number of Attackers	Traffic
KrebsOnSecurity	1.2 million	620 Gbps
OVH	145 thousands	1 Tbps
DYN	Millions	1.2 Tbps

At this level, every organization is vulnerable. The current number and the growth rate of new IoT devices show no evidence of slowing down. And finally, the nature of most IoT devices makes attacks on some of these new devices much more difficult to stop than attacks on conventional IT devices because they don't come with any kind of effective support.

Can We Fix These Problem?

Fixing these problems will be a challenge. Technically, some of the problems are straightforward to fix. For example:

- Do not allow default login/passwords, and require users to change them before the device is enabled.
- Do not have fixed unchangeable firmware login/passwords.
- Provide automatic updates of software.

Beyond that, matters get more difficult. How does the industry provide support for low-cost IoT devices? How long will they be supported? What happens when support ends; do the devices turn themselves off? How are attacks detected and contained? Harder still, how do we fix the currently very large deployed bases of IoT devices? Is anyone in the supply chain of the IoT devices concerned?

The economics of providing real security for IoT devices raises even more questions. Who is responsible when an IoT device is taken over by malware?

The user, the retail outlet where the device was purchased, the manufacturer, or the component vendors that the manufacturer used? How do you provide long-term support for a device that sells for less than $100 USD? What happens when support ends?

Unfortunately, there are many more questions than answers.

What Can We Do?

While there isn't a simple single solution to these problems, there are some things we can do. Possibilities range from what IoT device owners, companies that sell the devices, companies that design and build the devices, and the IoT Industry need to do.

The first and simplest is that people who own IoT devices should not run them with default passwords. That precept applies to all Internet devices. You always should use unique and hard-to-guess login usernames and passwords. You should use different passwords for each device. This common sense advice will greatly reduce the likelihood that IoT malware will infect your devices.

Note that using a password manager is recommended for IoT devices and, of course, all places where you have accounts on the Internet. These password managers allow you to use impossible-to-guess passwords. This approach is clearly much better than using "admin/admin."

The next step after setting new login/password information is to reboot the devices. The IoT malware lives in memory and will be erased if the device is rebooted. Then check to determine if the new login/passwords are still there. This step won't help with default unchangeable firmware accounts, but is still worth doing.

Companies that sell IoT devices need to ensure that the devices they sell are reasonably secure. This can include things like requiring login/passwords to be set upon installation, not having fixed unchangeable firmware passwords, and allowing users to get updates that fix known security problems. Retail channels like Amazon, Best Buy, NewEgg, etc., and companies that specialize in these devices like FLIR Systems may have the most leverage over companies that design and build these devices because the device manufacturers don't sell directly themselves. They all sell through well-known retail channels. These channels may also be liable if they are selling insecure IoT devices. Will someone file a class action suit against one of these companies? Time will tell.

Companies that design, build, and manufacture IoT devices should make their products more secure. They need to require that login/passwords account information needs to be configured as part of the installation procedure. They should also provide automatic security updates.

I am not sure that without any other motivation these companies will do this procedure, given how far in the channel they are from the end user. I think with serious encouragement from the retail channel, for example, telling them "we will stop selling your products until their security is improved," it will happen. There may even be a market for devices that are more secure than the others. On a positive note, it has been reported that the Chinese electronics firm Hangzhou Xiongmai Technology Co Ltd has initiated a recall of its IP cameras (webcams)[11]. It is hard to tell how effective this effort will be, but it clearly will not be as good as pushing software updates.

The IoT industry needs to develop an approach to make the IoT secure. If it continues on the current path, IoT is going to be a disaster for this industry. I don't think we are there yet, but the signs are not encouraging. We are not going to get the grand visions of IoT everywhere if people think the devices are insecure. The current IoT malware doesn't attack the owners of these devices, and they may not even be aware their devices are compromised, but this is only the beginning.

The potential for the collection of data on the users of these devices is staggering. If the device is compromised, its camera and sensors can be used against the user. Who wouldn't be susceptible to a ransom demand that threatened to share what you did in your house? While I am sure that most of us don't have anything serious to hide, would you want the world to see everything you have done or said in your house? We already have DDoS for hire firms, so what happens when we get surveillance companies hiring IoT surveillance for hire firms? It would certainly make divorce court proceedings more interesting.

One bit of good news based on these recent attacks, a lot of work is going on to create security frameworks for IoT devices. Organizations doing this experimentation include parts of the U.S. Government like the *National Institute of Standards and Technology* (NIST)[12] and the *Department of Homeland Security* (DHS)[13], and a variety of organizations including the *Internet Architecture Board* (IAB)[10], the *Broadband Internet Technical Advisory Group* (BITAG), the *GSM Alliance,* the *Industrial Internet Consortium,* and the *Open Web Application Security Project*. Bruce Schneier has compiled a good list of resources[14]. While this is all good news, it's not clear how these efforts will cause current and future IoT devices to change to be more secure. Some mix of consumer guidance, retail channel control, and/or government regulation may be needed.

Conclusion

The problem we have with the "Internet of Insecure Things" is only going to get worse for the immediate future. Many things need to happen to improve the situation, and it isn't going to happen quickly. I believe there are a lot of benefits if we can start building the Internet of Secure Things, but we have a long way to go. It's important that everyone involved take security more seriously. "Everyone" includes users, the retail channel, manufacturers of the devices, and the IoT industry. It will get better only if everyone gets involved.

References and Further Reading

[1] KrebsOnSecurity Blog: **https://krebsonsecurity.com**

[2] "KrebsOnSecurity Hit with Record DDoS," **https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/**

[3] "Akamai on the Record KrebsOnSecurity Attack," **https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/**

[4] "Protecting news from digital attacks," **https://projectshield.withgoogle.com/public/**

[5] The DDoS that didn't break the camel's VAC," **https://www.ovh.com/us/news/articles/a2367.the-ddos-that-didnt-break-the-camels-vac**

[6] Dyn Statement on 10/21/2016 DDoS Attack: **http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/**

[7] "Mapping Mirai: A Botnet Case Study," **https://www.malwaretech.com/2016/10/mapping-mirai-a-botnet-case-study.html**

[8] Malware Must Die: **http://blog.malwaremustdie.org**

[9] "A DDoS botnet aims IoT w/ IPv6 ready," **http://blog.malwaremustdie.org/2016/10/mmd-0059-2016-linuxirctelnet-new-ddos.html**

[10] Internet Architecture Board (IAB), "Report from the Internet of Things (IoT) Software Update (IoTSU) Workshop 2016)," **https://datatracker.ietf.org/doc/draft-iab-iotsu-workshop/**

[11] "China electronics firm to recall some U.S. products after hacking attack," **http://www.reuters.com/article/us-cyber-attacks-manufacturers-idUSKCN1200MS**

[12] "Framework for Improving Critical Infrastructure Cybersecurity," **https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf**

[13] US Department of Homeland Security, "Securing the Internet of Things," **https://www.dhs.gov/securingtheIoT**

[14] Bruce Schneier, "Security and Privacy Guidelines for the Internet of Things,"
https://www.schneier.com/blog/archives/2017/02/security_and_pr.html

[15] William Stallings, "The Internet of Things: Network and Security Architecture," *The Internet Protocol Journal,* Volume 18, No. 4, December 2015.

[16] Lake, D., Rayes, A., and Morrow, M., "The Internet of Things," *The Internet Protocol Journal,* Volume 15, No. 3, September 2012.

[17] Stankovic, J., "Research Directions for the Internet of Things," *Internet of Things Journal,* Volume 1, No. 1, 2014.

[18] Frahim, J., et al., "Securing the Internet of Things: A Proposed Framework," Cisco White Paper, March 2015.

[19] Gareth Corfield, "Fix crap Internet of Things security, booms Internet daddy Cerf," *The Register,* March 21, 2017,
https://www.theregister.co.uk/2017/03/21/vint_cerf_internet_things_security/

BOB HINDEN is a Check Point Fellow at Check Point Software, and co-chairs the IPv6 working group in the IETF. Hinden was the Chair of the Internet Society Board of Trustees from 2013 to 2016, and a member of the Board of Trustees from 2010–2016. Previously at Nokia, he was a Nokia Fellow, Chief Internet Technologist at Nokia Networks, and Chief Technical Officer (CTO) at the Nokia IP Routing Group. He was one of the early employees of Ipsilon Networks, Inc. Nokia acquired Ipsilon on December 31, 1997. Bob was previously employed at Sun Microsystems, where he was responsible for the Internet Engineering group that implemented internet protocols for Sun's operating systems. Prior to this position he worked at Bolt, Beranek, and Newman, Inc. on a variety of internetwork-related projects, including the first operational Internet router and one of the first TCP/IP implementations. Hinden was co-recipient of the 2008 *IEEE Internet Award* for pioneering work in the development of the first Internet routers. He has been active in the IETF since 1985 and is the author of 39 RFCs, including two April 1 RFCs. He served as the chair of the *IETF Administrative Oversight Committee* (IAOC) from 2009 through 2013. Before that he served on the *Internet Architecture Board* (IAB), was Area Director for Routing in the Internet Engineering Steering group from 1987 to 1994, and chaired the IPv6, Virtual Router Redundancy Protocol, Simple Internet Protocol Plus, IPAE, the IP over ATM, and the Open Routing working groups. He is also a member of the RFC Editorial Board and the RFC Series Oversight Committee. Hinden holds an B.S.E.E., and a M.S. in Computer Science from Union College, Schenectady, New York. E-mail: **bob.hinden@gmail.com**

DNS Privacy

by Geoff Huston and Joao Luis Silva Dama, APNIC

The *Domain Name System* (DNS) is normally a relatively open protocol that smears its data (which is your data and mine too!) far and wide. Little wonder that the DNS is used in many ways, not just as a mundane name resolution protocol, but as a data channel for surveillance and as a common means of implementing various forms of content access control.

But this situation is poised to change. The material released by Edward Snowden has sensitized us to the level of such activities that we have now become acutely aware that many of our Internet tools are just way too trusting, way too chatty, and way too easily subverted. First and foremost, in this collection of vulnerable Internet tools is the DNS.

A query made to the DNS is a precursor to almost every Internet transaction. Whether it's performing a search, downloading a web page, sending a mail message, opening a chat session, or even receiving an online advertisement, the DNS is often invoked as the first step. As users, we work in a symbolic world of readable names, such as **`facebook.com`** or **`netflix.com`**, whereas the underlying fabric of the Internet can send and receive packets only by using binary IP addresses rather than these symbolic names. So, we use the DNS to map from a name to an IP address. Thus, if you were able to look at a log of the DNS queries I've made in the last day or so, you may well be able to reconstruct my recent web browsing history, for example.

Obtaining a log of the DNS queries I make is perhaps the equivalent in terms of information content to obtaining a telephone log of called numbers from a previous generation of communications. A DNS transaction log may not provide information about the precise nature of the network transactions I've made, but it does record which sites I've been using. This information is often just good enough, as it's exactly what you would need to build a highly accurate profile of what I do on the Internet. It's not just national security bodies that have an interest in assembling such data logs. These days we see many systems that target individual users, and build a comprehensive profile of their needs and desires. The difference between an annoying advertisement and a timely helpful suggestion is just information about the user, and many companies assemble such profiles as part of their own commercial activities.

It's also true that the DNS is incredibly chatty. For example, to resolve a new name, such as **`www.example.com`**, a DNS resolver first asks the root name servers for the IP address of **`www.example.com`**. The root name servers would not be able to provide the answer, but they will respond with the authoritative name servers for the **`.com`** domain.

The resolver will then repeat this query relating to the IP address of the name **www.example.com.** to a **.com** name server, and once more the answer is an indirect one, indicating that while it does not know the answer, the list of name servers for the domain **example.com** should be queried. At this point the resolver can repeat the same query to a server that is authoritative for the **example.com** domain and probably receive an answer that contains the address of **www.example.com.**

But let's think about these DNS queries for a second. In this case, a root server, a **.com** server, and an **example.com** server are all now aware that I am "interested" in **www.example.com**, and they probably have stored a log of these queries. I have no idea if these logs are private or public. I have no idea how they are analyzed, and what inferences are drawn from this data.

It's possible that the data leakage is a little worse than described, because the application I am using, such as a browser, normally does not perform DNS name resolution itself. It passes the query to the platform operating system via a *gethostbyname()* call. The operating system platform also has an opportunity to log this query. The platform normally does not operate a standalone DNS resolver, and often is configured by the local network provider with DNS resolvers to use. So, my service provider may also be privy to all my DNS activity. But it need not stop there. My service provider might farm out its queries to a recursive forwarder, so that it can avoid the overheads of running a full DNS resolver.

Normally such forms of query indirection imply a loss of attribution, as such forwarded queries do not have any of my identifying details. Unless of course the resolver uses the EDNS0 *Client Subnet Option*[1], in which case the forwarded queries still contain some critical details of my network, and, by inference, me as well.

All of these DNS queries can represent a lot of information, even in these days of data intensity. Back in April 2015 Google reported that its public DNS servers deliver some 400 billion responses per day[2], and it appears that Google resolves some 12% of the total DNS load[3], so there were some 3 trillion DNS queries per day at that time. It can only be larger today.

Not only is the DNS a chatty protocol that gratuitously sprays out information about user behaviours, it does so in an entirely open manner. DNS queries and their responses are unencrypted, and are sitting on port 53 in *User Datagram Protocol* (UDP) and *Transmission Control Protocol* (TCP). Because they are open and unencrypted, DNS queries can be easy to intercept, and, if *Domain Name System Security Extensions* (DNSSEC) is not used, false responses can be inserted back into the data stream, and the client may be none the wiser. In some countries, DNS substitution appears to be relatively commonplace[4].

Other countries have turned to DNS interception and blocking in response to problems associated with overloading IP addresses with virtual web hosting[5]. Little wonder that many users have tried to get around these local efforts to block access by using a third-party DNS resolver, and the use of Google's Public DNS resolver is a common response to such local efforts to interfere in the DNS. Indeed, so common is this response that now the local DNS blocking measures appear to include intercepting access to Google's DNS service as well![6]

DNS privacy has been a matter of some interest to the *Internet Engineering Task Force* (IETF), and changes are being proposed to the DNS protocol that would make it far harder to be used as a snooper's and censor's tool of choice.

What is going on to improve this situation and introduce aspects of privacy into the DNS?

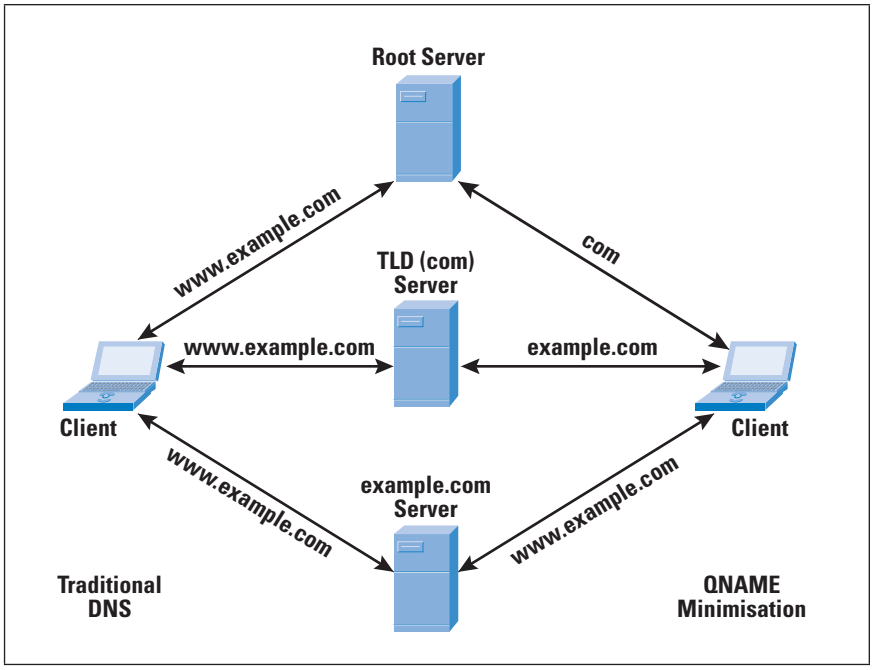
QNAME Minimisation

The IETF *DNS Operations* Working Group has been performing what has been called query name minimisation in the DNS, resulting in a specification for *QNAME Minimisation*.[7] To quote from this document: "QNAME minimisation follows the principle [that] the less data you send out, the fewer privacy problems you have."

In the previous example, the query to the root servers for the A record for `www.example.com` has two elements of gratuitous information: the fully qualified domain name and the query type. A more targeted query that does not gratuitously leak extraneous information is a query directed to the root name servers for the name server records for the `.com` domain. Similarly, the `.com` name servers would be queried simply for the name servers of `example.com` and so on (see Figure 1).

In general, this approach is no less efficient than using a full query name at every point, and is equally capable of using cached information. The technique has exposed some inconsistencies with the handling of so-called *empty nonterminal* domain names, but the approach can be implemented in a robust manner, and it is a solid step in plugging a gratuitous information leak. It appears that the recently announced Knot DNS resolver from the Czech domain provider CZ.NIC is one of the first DNS resolvers to implement QNAME Minimisation (`https://www.knot-resolver.cz`).

Figure 1: The Intended Operation of QNAME Minimisation



DNS and TLS

However, QNAME Minimisation is only part of the privacy story. The open nature of DNS queries makes third-party monitoring, interception, and substitution incredibly easy, it appears. The *DNS PRIVate Exchange* (DPRIVE) Working Group of the IETF has been working on this topic, looking at ways for the DNS query and response interaction between a DNS client and a DNS resolver to be protected in some manner.

There are two parts to this work. Firstly, to ensure that the response you receive is a response from the DNS resolver that you intended to ask, and secondly, to ensure that the query you pass to your DNS resolver is not readily readable by anyone other than the addressed DNS resolver.

One issue here is whether to try to secure the current UDP-based resolution protocol, or head to a TCP-based approach where solutions already abound, typically based on *Transport Layer Security* (TLS). TLS encrypts the conversation between a client and a server using session keys that are generated based on random seed values coupled with public/private key pairs. If you have a way of associating a service name (such as its DNS name) with a public key (as would be the case with a conventional DNS Name public key certificate), and a way of validating that key (using conventional name certificate validation), then you can achieve both of these objectives. The remote server has demonstrated to you that it has knowledge of the private key associated with the DNS name of the service, which is theoretically known only to the server and no one else.

By encrypting your session with a session key that is based in part on this private key, the content of the data exchange should be protected from onlookers and potential interceptors in a man-in-the-middle attack.

But there is a problem here in terms of the transport protocol of choice. TLS conventionally requires a reliable transport channel, such as provided by TCP, and as such cannot be used directly to secure datagram traffic as used by UDP. However, the DNS has been heavily reliant on the use of UDP as a means of supporting speed and scalability in the DNS. So, from some perspectives, DNS-over-TLS-over-TCP is not seen as the optimal response to the problem. TCP attempts to ensure sequenced delivery, and in a message-oriented application, the loss of a message in TCP holds up the delivery of all subsequent messages until TCP can correct the data loss and deliver the lost message. This TCP "head-of-line blocking" can pose unacceptable overheads when using TCP to carry the datagram-like message payloads of the DNS. DNS over *Internet Protocol Security* (IPsec) could be seen as offering a cleaner fit when looking at securing a UDP-based application, but IPsec is a kernel function rather than an application module, and its semantics apply at the IP layer rather than as an attribute of the transport protocol. This reality makes it challenging to incorporate IPsec into an application and operate the cryptographic functions in user space, as happens with DNS name resolution.

One of the consequent investigations has been to see if the functionality of TLS could be mapped into a datagram transport environment. Out of this consideration has come a new protocol, *Datagram Transport Layer Security* (DTLS), which is an adaptation of the TLS function that can present to the application a datagram-like delivery function that does not require reliable transport services. DTLS is intended to be able to recover from packet loss and reordering, but it would be intolerant of UDP packet fragmentation[8]. Its design is modelled upon TLS 1.2 and intends to use some explicit additional features that allow TLS to function over a datagram transport as distinct from a reliable stream transport. DTLS intends to minimise the impact of the use of TLS on the DNS experience, particularly when compared to DNS-over-TLS-over-TCP. The major change envisaged would be to require an initial DTLS handshake to set up a shared encryption state, and the use of cookies to reuse that state across multiple individual response/query interactions.

One of the main features of the current DNS protocol when used over UDP is how little shared state overhead each individual transaction incurs, resulting in a highly responsive and capable service. DNS over DTLS attempts, as far as possible, to preserve this simple query/response datagram exchange model but does so in a manner where the client is using an encryption based on the validated credentials offered by the server. The current state of play of this specification is described in an Internet Draft working document.[9]

DTLS is intolerant of IP fragmentation, so the operation of DNS over DTLS is similar in design to the use of the *Truncated* bit in DNS over UDP as a signal to the client to repeat the query using TCP. Here the intended operation is that if a DNS-over-DTLS server has a response that is greater than the local Path *Maximum Transmission Unit* (MTU) estimate, then the server should set the Truncated bit in its response, and this response is to be interpreted by the client as a signal that the client should repeat the query using DNS-over-TLS-over-TCP, in a manner analogous to the current use of the Truncated bit to signal to a client to repeat the query using TCP rather than UDP. However, it needs to be noted that at this point DTLS remains a design exercise, and it may be some time before implementations of this specification are available for general use by end-user DNS libraries, recursive resolvers, and servers.

The other option here is to absorb the TCP overheads into the solution and just use conventional TLS, which is a TCP service. Much has been said on the use of TCP as a mainstream transport protocol for DNS, as distinct from its current intended role as a backup to UDP for large responses. It has been argued that the TCP connection state overheads of the servers seriously impair their ability to handle large query loads, and the additional overhead of the protocol handshake would negatively affect the user experience. On the other hand, it is argued that already the web is being used overwhelmingly as a short transaction service, and web servers appear to withstand the imposed load. It is also noted that the use of TCP is an effective measure against various forms of abuse that rely upon the ability to perform source address spoofing in UDP.

The specification for *DNS over Transport Layer Security*[10] is a relatively straightforward description, in that the transport service offered by TLS is effectively the same as that offered by TCP, but running the listener of the server at TCP port 853, rather than port 443. There is perhaps one change here, and that is a suggestion for TLS session reuse: "In order to minimize latency, clients SHOULD pipeline multiple queries over a [single] TLS session." For transactions between a client and a recursive resolver, the suggestion for session reuse makes some sense. For transactions between a client and authoritative name servers where the client is itself performing DNS resolution, this choice may not be so readily achievable. The choice of a distinguished TCP port is also interesting. If you wanted the secure channel DNS traffic to merge into all other traffic and pose a challenge to attempts to block this service, the temptation to use port 443 for DNS over TLS would be overwhelming (at least for me!). More information on the current state of clients and servers that support DNS over TLS can be found at:
`https://portal.sinodun.com/wiki/display/TDNS`.

Secure DNS over JSON

Last, but not least, there is the option to use an entirely different data encoding protocol, and here a recently announced service from Google is relevant. The server at **`https://dns.google.com`** performs a resolution function over TLS using port 443 with the results passed back as a *JavaScript Object Notation* (JSON) data structure. This function can readily be transformed into an alternative form of *gethostbyname()* by the application's substituting a web object retrieval for a conventional DNS query. This substitution offers the caller some level of privacy from third-party inspection and potential intrusion and censorship, although it's unclear precisely what "privacy" means when you are sharing your DNS activity with Google!

Example script:

```
#! /usr/bin/env python
import json, requests

url = "https://dns.google.com/resolve"
params = dict(
    name='www.potaroo.net',
    type='A',
    dnssec='true'
)

resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print data[u'Answer'][0][u'data']
```

Application-Level DNS

Concerns about data leakage is not limited to external forms of surveillance and interception. An appropriately paranoid application would not use the DNS resolution service of the underlying host platform, because that would release the name queries of the application into an uncontrolled environment where it may be logged and accessed by the platform and other applications.

One response to this potential for uncontrolled leakage of information is for the application to assume a greater role in performing DNS resolution. The simple approach is for the application to outsource this role to a trusted agent, and do so over a secured channel, which is where secure DNS over JSON comes in. In this case, the application is not performing DNS resolution and validation itself, but in creating a secured channel to Google's resolution service across a TLS connection the application can obtain some level of assurance that it is not performing a local leak of DNS information, and that with DNSSEC validation enabled, the responses it receives have some level of assurance that they are genuine, assuming that the name being resolved is itself DNSSEC-signed.

But perhaps even that is too much outsourced trust. Another approach is being constructed by the *GetDNS* project.[11] In this case, it's not a secure channel to a recursive resolver that will resolve the application queries. GetDNS provides the application with a local validating resolver as a set of library calls. This project currently supports DNS over TLS. The GetDNS project operates as an open source project, and the GetDNS project page contains pointers to the code. A web application is built into the API, and a portal to a resolver implemented in this manner can be found at:
`https://getdnsapi.net/query`

This approach of pulling the DNS resolution function potentially all the way back into the application has some interesting trade-offs. The queries being made now have a source address of the local host, so the data that is leaked through the DNS queries can identify the local host. If an authoritative name server does not support a secure channel for queries using DNS over TLS, then the API will necessarily use an open unencrypted channel (at this stage the DNS over DTLS is not included in the GetDNS code base, but if someone wants to submit code …).

On the other hand, the DNSSEC validation function can be performed locally as well by a GetDNS instance, so that the application is not forced to trust the authenticity of a bit flag in the response from a remote resolver. This way the application has direct control of the validation function, and direct knowledge of its outcome.

Between these two approaches there are further apparent trade-offs.

Making queries via a secure channel to a busy recursive resolver—and Google's Public DNS is about as busy as a DNS resolver can be—means that it is possible, to some extent, to hide behind the cache of such busy resolvers. As long as you are comfortable with sharing your DNS queries with Google, then to some extent you can use secured access to a DNS recursive resolver that intends to operate with integrity, accuracy, and completeness. The secure channel is far harder to subvert and more resistant to efforts to eavesdrop on the query stream.

If you are uncomfortable with this approach, then another option is to pull the name resolution function back into your platform and even back into the application itself using a framework such as GetDNS. The extent to which your queries may be readily visible to third parties, and the extent to which your query stream may be subverted in various ways, now depends on the capabilities of the authoritative name servers. Without name server support for DNS over TLS and possibly also potential future support for DNS over DTLS, and without DNSSEC signed zones, the local DNS resolver may still be misled in ways that may not be readily detected.

In this case, the local resolver is powerless to fix this problem, because the privacy and protection mechanisms are now in the hands of the authoritative name servers and the zone admins that are queried by the local resolver.

With DNS privacy, there is no such thing as a free lunch! But maybe in QNAME Minimisation there is the possibility of a much cheaper lunch. The queries are much the same as before, but the impact is that the query name is only progressively revealed to the name servers that are authoritative for the domain being queried.

However, an equal cause for concern is that the queries and responses are open and susceptible to eavesdropping. Here the lunch is definitely more expensive! The measures to introduce a secure channel for DNS queries and responses take a simple open query/response stateless protocol and introduce state by way of session establishment and crypto state establishment. The overheads of such additional state can be many packets and the imposition of additional Round Trip Time intervals. It is also true that no DNS privacy solution is absolute in these frameworks. While it is possible to set up a secure and encrypted channel to a recursive resolver, the implication is that the recursive resolver is privy to the query stream, even if the local network infrastructure cannot directly eavesdrop on the queries.

Alternatively, the user application can operate as a resolver itself, and attempt to direct queries to authoritative servers over a secured channel, but the authoritative servers now have visibility on your identity, and they also have to cope with a dramatic change in the query load. The authoritative name server would no longer be able to rely on recursive resolvers to absorb many of the queries, so they would presumably be exposed to a far larger query load. In addition, they would need to maintain a significant state overhead to support secured channels to these end application-based name resolvers. This really does not look like a likely scenario. As a result, it appears, at least for the moment, that this work on secured transport channels is likely to be a measure used between end-user applications and recursive resolvers.

What Does All This Mean?

While today the open nature of DNS queries makes third-party monitoring, interception, and substitution incredibly easy, there are now some grounds to be optimistic and start to contemplate a DNS environment that preserves privacy and integrity.

By performing QNAME Minimisation it is possible to radically reduce the level of leaked information coming from the DNS, and by wrapping up DNS queries and responses in a secured channel it is no longer trivial for third parties to monitor and intercept DNS queries and their responses on the wire.

If applications made use of services that push local DNS query traffic into encrypted TLS sessions with recursive resolvers, such as the service Google offers, the result would be that much of today's visible DNS would disappear from view. Not only that, but it would make the existing practices of selective local inspection and intervention in the DNS resolution process far more challenging, if not infeasible. It may be even better if authoritative name servers were to also support queries over TLS and DTLS, allowing a local host to take over the resolution function and still use encrypted query traffic services.

If this scenario were to be coupled with widespread use of DNSSEC, then it would be a somewhat different Internet from the one we have today. It's pretty obvious that national online censorship efforts will continue, and online monitoring and surveillance will also continue. But the ability to coopt the DNS into the role of an exceptionally cheap and simple means to achieve these ends will cease at some time if we collectively choose to head down this path for adding privacy and security to the DNS.

References

[1] Wilmer van der Gaast, Carlo Contavalli, and Warren Kumari, "Client Subnet in DNS Queries," RFC 7871, May 2016.

[2] Google Blog, "Google Public DNS and Location-Sensitive DNS Responses," December 2014,
https://webmasters.googleblog.com/2014/12/google-public-dns-and-location.html

[3] APNIC Labs, "DNSSEC Validation Rate by Country,"
https://stats.labs.apnic.net/dnssec

[4] Byron Ellacott and Geoff Huston, "Facebook and the GFW," August 2013,
http://www.potaroo.net/presentations/2013-08-29-facebook.pdf

[5] Geoff Huston, "The Company You Keep," *The ISP Column,* June 2013.
http://www.potaroo.net/ispcol/2013-06/company.html

[6] Babak Farrokhi, "Operator Level DNS Hijacking," RIPE Labs, July 2016.
https://labs.ripe.net/Members/babak_farrokhi/operator-level-dns-redirection

[7] Stephane Bortzmeyer, "DNS Query Name Minimisation to Improve Privacy," RFC 7816, March 2016.

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

[9] Dan Wing, Tirumaleswar Reddy, and Prashanth Patil, "Specification for DNS over Datagram Transport Layer Security (DTLS)," Internet Draft, Work in Progress, December 2016, **draft-ietf-dprive-dnsodtls-15**.

[10] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, May 2016.

[11] GetDNS Project: **https://getdnsapi.net**

GEOFF HUSTON, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990s. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001. At various times Geoff has worked as an Internet researcher, an ISP systems architect, and a network operator. E-mail: **gih@apnic.net**

JOAO LUIS SILVA DAMAS is currently Senior Researcher at APNIC. Joao was also co-founder of Hivecast Inc, a DNS services company later sold to Dyn. Previously he worked at ISC (Internet Systems Consortium) as CTO overseeing technical developments. Earlier, he served as CTO at RIPE NCC and later founded Bond Internet Systems as a consulting and research company. For around 7 years he organised the RIPE plenary program and launched the current RIPE program committee. In 2008, together with colleagues, Joao launched ESNOG to bring together Spanish ISPs to interact with each other, an activity that continues to this day.
E-mail: **joao@bondis.org**

Fragments

ICANN Launches Testing Platform for the KSK Rollover

The *Internet Corporation for Assigned Names and Numbers* (ICANN) is offering a testing platform for network operators and other interested parties to confirm that their systems can handle the automated update process for the upcoming Root Zone *Domain Name Systems Security Extensions* (DNSSEC) *Key Signing Key* (KSK) rollover[1, 2]. The KSK rollover is currently scheduled for October 11, 2017.

"Currently, seven hundred and fifty million people are using DNSSEC-validating resolvers that could be affected by the KSK rollover," said ICANN's Vice President of Research, Matt Larson. "The testing platform is an easy way for operators to confirm that their infrastructure supports the ability to handle the rollover without manual intervention."

Internet service providers, network operators and others who have enabled DNSSEC validation must update their systems with the new KSK. This can be done in one of two ways:

- An operator can configure a new trust anchor *manually* by obtaining the new root zone KSK from the *Internet Assigned Numbers Authority* (IANA) website at:
  `https://www.iana.org/dnssec/files`
- An operator can enable a feature available in many validating resolvers that *automatically* detects and configures a new root zone KSK as a trust anchor, in which case they need take no action.

Check to see if your systems are ready by visiting:
`go.icann.org/KSKtest`

The KSK has been widely distributed and configured by every operator performing DNSSEC validation. If the validating resolvers using DNSSEC do not have the new key when the KSK is rolled, end users relying on those resolvers will encounter errors and be unable to access the Internet. A careful and coordinated effort is required to ensure that the update does not interfere with normal operations.

More information is available at **`www.icann.org/kskroll`**

[1] George Michaelson, Patrick Wallström, Roy Arends, and Geoff Huston, "Rolling Over DNSSEC Keys," *The Internet Protocol Journal*, Volume 13, No 1, March 2010.

[2] ICANN Blog, "The Problem with 'The Seven Keys,'" February 13, 2017. **`https://www.icann.org/news/blog/the-problem-with-the-seven-keys`**

Follow us on Twitter and Facebook

`@protocoljournal`

`https://www.facebook.com/newipj`

Thank You!

Publication of IPJ is made possible by organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol. The following individuals have provided support to IPJ. You can join them by visiting **http://tinyurl.com/IPJ-donate**

Fabrizio Accatino
Scott Aitken
Antonio Cuñat Alario
Matteo D'Ambrosio
Jens Andersson
Danish Ansari
David Atkins
Jaime Badua
John Bigrow
Axel Boeger
Kevin Breit
Ilia Bromberg
Christophe Brun
Gareth Bryan
Stefan Buckmann
Scott Burleigh
Jon Harald Bøvre
Olivier Cahagne
Roberto Canonico
Lj Cemeras
Dave Chapman
Stefanos Charchalakis
Greg Chisholm
Narelle Clark
Steve Corbató
Brian Courtney
Dave Crocker
Kevin Croes
John Curran
Morgan Davis
Freek Dijkstra
Geert Van Dijk
Ernesto Doelling
Karlheinz Dölger
Andrew Dul
Holger Durer
Peter Robert Egli
George Ehlers
Peter Eisses
Torbjörn Eklöv
ERNW GmbH
ESdatCo
Steve Esquivel
Mikhail Evstiounin
Paul Ferguson
Christopher Forsyth
Craig Fox
Tomislav Futivic
Edward Gallagher
Andrew Gallo
Chris Gamboni
Xosé Bravo Garcia
Kevin Gee
Serge Van Ginderachter
Greg Goddard
Octavio Alfageme Gorostiaga
Barry Greene
Martijn Groenleer
Geert Jan de Groot
Gulf Coast Shots
Sheryll de Guzman
Martin Hannigan
John Hardin
Edward Hauser
Headcrafts SRLS
Edward Hotard
Bill Huber
Hagen Hultzsch
Karsten Iwen
David Jaffe
Dennis Jennings
Jim Johnston
Jonatan Jonasson
Daniel Jones
Gary Jones
Amar Joshi
Merike Kaeo
David Kekar
Shan Ali Khan
Nabeel Khatri
Henry Kluge
Carsten Koempe
Alexader Kogan
Mathias Körber
John Kristoff
Terje Krogdahl
Bobby Krupczak
Warren Kumari
Darrell Lack
Yan Landriault
Markus Langenmair
Fred Langham
Richard Lamb
Tracy LaQuey Parker
Robert Lewis
Sergio Loreti
Guillermo a Loyola
Hannes Lubich
Dan Lynch
Miroslav Madic´
Alexis Madriz
Carl Malamud
Michael Malik
Yogesh Mangar
Bill Manning

Harold March
David Martin
Timothy Martin
Gabriel Marroquin
Carles Mateu
Juan Jose Marin Martinez
Brian McCullough
Joe McEachern
Carsten Melberg
Kevin Menezes
Bart Jan Menkveld
William Mills
Thomas Mino
Mohammad Moghaddas
Charles Monson
Andrea Montefusco
Fernando Montenegro
Tariq Mustafa
Stuart Nadin
Mazdak Rajabi Nasab
Krishna Natarajan
Darryl Newman
Ovidiu Obersterescu
Mike O'Connor
Carlos Astor Araujo Palmeira
Alexis Panagopoulos
Manuel Uruena Pascual
Ricardo Patara
Dipesh Patel
Alex Parkinson
Craig Partridge
Dan Paynter
Leif-Eric Pedersen
Juan Pena
Chris Perkins
Rob Pirnie
Blahoslav Popela
Tim Pozar
David Raistrick
Priyan R Rajeevan
Paul Rathbone
Bill Reid
Justin Richards
Mark Risinger
Ron Rockrohr
Carlos Rodrigues
William Ross
Boudhayan Roychowdhury
Carlos Rubio
RustedMusic
Babak Saberi
George Sadowsky
Scott Sandefur
Arturas Satkovskis
Phil Scarr
Jeroen Van Ingen Schenau
Carsten Scherb
Roger Schwartz
SeenThere
Scott Seifel
Yury Shefer
Yaron Sheffer
Tj Shumway
Jeffrey Sicuranza
Thorsten Sideboard
Geoff Sisson
Helge Skrivervik
Darren Sleeth
Mark Smith
Job Snijders
Ignacio Soto Campos
Peter Spekreijse
Thayumanavan Sridhar
Matthew Stenberg
Adrian Stevens
Clinton Stevens
Viktor Sudakov
Edward-W. Suor
Vincent Surillo
Roman Tarasov
David Theese
Sandro Tumini
Phil Tweedie
Steve Ulrich
Unitek Engineering AG
John Urbanek
Martin Urwaleck
Betsy Vanderpool
Surendran Vangadasalam
Alejandro Vennera
Luca Ventura
Tom Vest
Dario Vitali
Randy Watts
Andrew Webster
Tim Weil
Jd Wegner
Rick Wesson
Peter Whimp
Jurrien Wijlhuizen
Pindar Wong
Bernd Zeimetz

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ 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. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- 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, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud 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, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at **ole@protocoljournal.org** or **olejacobsen@me.com**

Supporters and Sponsors

Supporters

Internet Society

CISCO

Diamond Sponsors

Google

Ruby Sponsor

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

Sapphire Sponsors

Your logo here!

Emerald Sponsors

Afilias

apia

APNIC

CISCO

COMCAST

DE-CIX

EQUINIX

ICANN

JUNIPER NETWORKS

lacnic

LinkedIn

NSRC
Network Startup Resource Center

NTT Communications

TEAM CYMRU

VERISIGN

WIDE PROJECT

Corporate Subscriptions

amsix
amsterdam internet exchange

DNS-OARC

InterWorking Labs®

Limelight NETWORKS

netnod

qa|cafe

SIDN

For more information about sponsorship, please contact `sponsor@protocoljournal.org`

The Internet Protocol Journal
NMS
535 Brennan Street
San Jose, CA 95131

ADDRESS SERVICE REQUESTED

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org

Printed in the USA on recycled paper.