The Internet Protocol Journal

December 2022 Volume 25, Number 3

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

Although most of the core protocols of the Internet have remained unchanged for many decades, there is still active work in the *Internet Engineering Task Force* (IETF) and elsewhere to enhance, improve, and even replace some functions provided by the protocols with respect to design and with focus on operational best practice. In this issue we will explore two areas where such work is in progress, namely transport and the *Domain Name System* (DNS).

In our first article, Geoff Huston compares the *Transmission Control Protocol* (TCP) to QUIC. QUIC has seen rapid deployment in the Internet largely due to its improved performance and extensibility, as well as privacy and security. Geoff predicts that QUIC may some day replace TCP as the major transport protocol in the Internet. The IETF has a working group dedicated to QUIC.

Another very active work area in the IETF focuses on the DNS. One topic of interest in the *DNS Operations* (DNSOP) working group is *Minimized DNS Resolution*. In our second article, Burton Kaliski, Jr. describes four different approaches to minimization: *Qname Minimization*, *NXDOMAIN Cut Processing*, *Aggressive DNSSEC Caching*, and *Local Root*. All of these approaches have been documented in RFCs and are in various stages of deployment across the Internet.

Our final article, by Adiel Akplogan, is an overview of *Knowledge-Sharing and Instantiating Norms for Domain Name System and Naming Security* (KINDNS) [pronounced "kindness"], an initiative launched by the *Internet Corporation for Assigned Names and Numbers* (ICANN) to promote DNS security and best practices.

I was very pleased to learn that the *2022 Jonathan B. Postel Service Award* was awarded to my friend and mentor George Sadowsky for his work on the Internet Society's *Developing Countries Workshops* in the 1990s. The tradition of training network engineers on all aspects of Internet technology continues at numerous conferences around the world to this day, most notably at events hosted by various *Network Operator Groups* (NOGs) and regional events such as APRICOT. For more details on the award, visit: **https://tinyurl.com/Postel2022**

As always, we welcome your feedback and suggestions on anything you read in this journal. Letters to the Editor may be edited for clarity and length and can be sent to **ipj@protocoljournal.org**

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

Comparing TCP and QUIC

by Geoff Huston, APNIC

A common view out there is that the QUIC transport protocol[0, 4] is just another refinement to the original *Transmission Control Protocol* (TCP) transport protocol[1, 2]. I find it hard to agree with this sentiment, because for me QUIC represents a significant shift in the set of transport capabilities available to applications in terms of communication privacy, session-control integrity, and flexibility. QUIC embodies a different communications model that makes it intrinsically useful to many more forms of application behaviours. Oh, yes. It's also faster than TCP! In my opinion it's likely that over time QUIC will replace TCP in the public Internet. So, for me QUIC is a lot more than just a few tweaks to TCP. Here we will describe both TCP and QUIC and look at the changes that QUIC has brought to the transport table.

However, we should first do a brief recap of TCP.

What Is TCP?

TCP is the embodiment of the end-to-end principle in the overall Internet architecture. All the functionality required to take a simple base of datagram delivery and impose upon this model an end-to-end signalling regime that implements reliability, sequencing, adaptive flow control, and streaming is embedded within the TCP protocol.

TCP is a *bilateral full-duplex* protocol. That means that TCP is a two-party communications protocol that supports both parties simultaneously, sending and receiving data within the context of a single TCP connection. Rather than impose a state within the network to support the connection, TCP uses synchronized state between the two end points, and much of the protocol design ensures that each local state transition is communicated to, and acknowledged by, the remote party without any mediation by the network whatsoever.

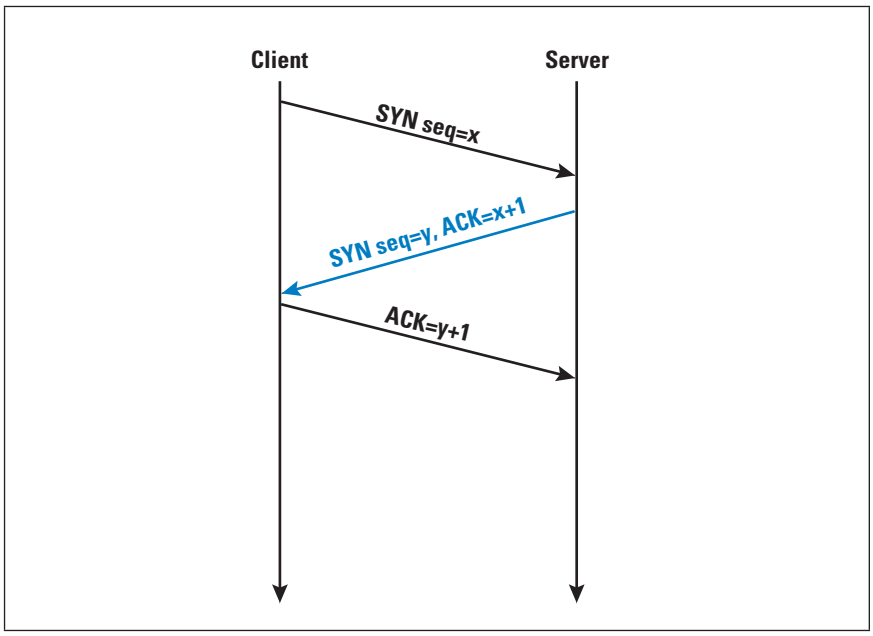
TCP is a *stream* protocol. The receiver sees the stream of data that the sender generates is in precisely the same order as the sender generated. TCP is a true streaming protocol, and application-level network operations are not transparent. Other transport protocols have explicitly encapsulated each application transaction; for every sender's *write*, there must be a matching receiver's *read*. In this manner, the application-derived segmentation of the data stream into a logical record structure is preserved across the communication. TCP does not explicitly preserve such an implicit structure of the data, so there is no explicit pairing between *write* and *read* operations within the network protocol. For example, a TCP application may write three data blocks in sequence into the network connection, which the remote reader may collect in a single read operation. It is left to the application to mark the stream with its own record boundaries, if such boundaries exist in the data.

A rudimentary level of stream formatting is permitted within TCP through the concept of *urgent data* in which the sender can mark the end of a data segment that the application wants to bring to the attention of the receiver. The TCP data segment that carries the final byte of the urgent data segment can mark this data point, and the TCP receiving process has the responsibility to pass this mark to the receiving application.

The hosts at both ends identify the TCP connection by a 5-tuple of protocol identifier, source IP address, source port, destination IP address, and destination port.

The setup of a TCP connection requires a *three-way handshake*, ensuring that both sides of the connection have an unambiguous understanding of the byte-sequence values of the remote side. The operation of the connection setup is as follows: The local system sends an initial sequence number to the remote-end port using a SYN packet. The remote system responds with an *acknowledgement* (ACK) of the initial sequence number and the remote end's initial sequence number in a response SYN packet. The local end responds with an ACK of this remote sequence number. These handshake packets are conventionally TCP packets without any data payload. At this point, TCP shifts into a reliable data flow-control mode of operation. (Figure 1)

Figure 1: TCP 3-way Handshake



TCP is a *sliding window* protocol. The data stream is a sequence of numbered bytes. The sender retains a copy of all sent but as yet unacknowledged data in a local send buffer. When a receiver receives a data segment whose starting sequence number is the next expected data segment, it will send an ACK back to the sender with the sequence number of the end of the received data segment. This process allows the sender to discard all data whose sequence number is less than this received ACK sequence in the local send buffer and advance the send window.

When the received data is out of order, it will send an ACK back to the sender with the sequence number of the last in-order received data. In addition, the ACK message includes the size of the receiver's available buffer size (*receive window*). The volume of unacknowledged data must be no larger than this receiver window size. The overall constraint is that at all points in time the sender should ensure that the volume of unacknowledged data in flight in the network is the smaller of the advertised receive window size and the total capacity of the local send buffer.

TCP is an *ACK-clocked* flow-control protocol, in that within a static lossless mode of operation each received ACK packet indicates that a certain volume of data has been received at the receiver end (and hence has been removed from the network), and this clocking is accompanied by an advertised receive window that then permits the sender to inject the same volume of data into the network as the receiver has removed. Hence, the sending rate is governed by the received ACK rate.

However, TCP is not necessarily aware of the available path capacity of the network, and it must implement a control algorithm at the sending end that attempts to establish a dynamic equilibrium between the flow volume of the TCP session and all other concurrent TCP sessions that have path segments in common with this session. The mode of operation of this flow control is not fixed in the TCP specification, and numerous flow-control algorithms are in use. Many of these control algorithms use an induced instability in TCP through an approach of slow inflation of the sending window for each received ACK, and a rapid drop of the sending window in response to an indication of packet drop (3 duplicate ACKs). This process of sending rate inflation will stop when the send buffer is full, indicating that the sender cannot store any more sent data and must await ACKs before sending more data (send buffer rate limited). It will also stop sending rate inflation when the network cannot accept any further data in flight as the buffers of the network are already full, so further sent data will cause packet loss, which will be signalled back to the sender by duplicate ACKs.

This process has many outcomes relevant to service quality. First, TCP behaves adaptively rather than predictively. The flow-control algorithms are intended to increase the data-flow rate to fill all available network path capacity but also quickly back off if the available capacity changes because of network congestion or if a dynamic change occurs in the end-to-end network path that reduces this available capacity. Second, a single TCP flow across an otherwise idle network attempts to fill the network path with data, optimizing the flow rate (as long as the send buffer is larger than the network flow capacity). If a second TCP flow opens up across the same path, the two flow-control algorithms will interact so that both flows will stabilize to use approximately half of the available capacity per flow. More generally, TCP attempts to behave fairly, in that when multiple TCP flows are present the TCP algorithm is intended to share the network resource evenly across all active flows.

A design tension always exists between the efficiency of network use and enforcing predictable session performance. With TCP, you do not necessarily have predictable throughput but gain a highly utilized and efficient network.

TCP and TLS

Transport Layer Security (TLS)[3] is handled as a further layer of indirection. When the TCP 3-way handshake is complete, the parties enter a TLS negotiation phase to allow authentication of the remote end of the connection, and to establish a session key that is used to manage the encryption of the session data.

TLS commences with an exchange of credentials. In version 1.3 of TLS (the latest version of this protocol), the client sends a *client hello* message that includes the TLS version the client supports, the cipher suites supported, the name of the service, and a string of random bytes known as the *client random*. The server responds with a *server hello* message that contains the public key certificate of the server, the *server random*, the chosen cipher suite, and a digital signature of the hello messages. Both ends now know each other's random values and the chosen cipher suite, so both can generate a master secret for session encryption. The client sends a *finished* message to indicate that the secure symmetric session key is ready for use (Figure 2).

Earlier versions of TLS used additional packets in the hello exchange that increased the time to complete the TLS handshake.

QUIC

We can now move on to QUIC. The QUIC transport protocol[4] was apparently designed to address several issues with TCP and TLS, and in particular to improve the transport performance for encrypted traffic with faster session setup, and to allow for further evolution of transport mechanisms and explicitly avoid the emerging TCP ossification within the network.

It is a grossly inaccurate simplification, but at its simplest level QUIC is simply TCP encapsulated and encrypted in a *User Datagram Protocol* (UDP) payload. To the external network QUIC has the appearance of a bidirectional UDP packet sequence where the UDP payload is concealed. To the endpoints you can use QUIC as a reliable full-duplex data-flow protocol. Even at this level, QUIC has numerous advantages over TCP. The first lies in the deliberate concealing of the transport flow-control parameters from the network. The practice of deploying network middleware that rewrites TCP flow-control values to impair the behaviour of the application has not enjoyed widespread support from the application layer, and hiding these flow-control parameters from the network certainly prevents this practice. Second, it can allow the shift of responsibility for providing the transport protocol from the platform to the application. A tension between the application and the platform is longstanding.

Changes to kernel-level TCP are performed via updates to the platform software, and often applications have to wait for the platform to make changes before they can take advantage of the change. For example, if an application wanted to use the TCP *Bottleneck Bandwidth and Round-trip Propagation Time* (BBR) flow-control algorithm, then it would need to wait for a platform to integrate an implementation of the algorithm. By using a basic UDP interface to the transport services of the platform, you can lift the entire flow-control and encryption service into the application itself, if so desired. You may experience some performance penalty of shifting the transport code from the kernel to user space, but in return the application regains complete control of the transport service and allows it to operate in a mode that is not only independent of the platform, but also hidden from the platform. This shift gives the application environment greater levels of control and agility.

However, QUIC does a lot more than just wrapping up TCP in UDP, so let's look at the QUIC protocol in a little more detail.

QUIC Connections

A QUIC *connection* is a shared state between a single client and a single server. QUIC uses the combination of two numbers, one selected by each end, to form a pair of connection IDs. This pair of IDs acts as a persistent identity for the QUIC session, which is used to ensure that changes in addressing at lower protocol layers (addresses or ports) will not cause delivery of packets to a wrong recipient on the end host.

The primary function of a connection ID is to ensure that changes in addressing at lower protocol layers (IP source address and UDP source port numbers) do not cause packets for a QUIC connection to be dropped when the external IP address of an endpoint changes. Each endpoint selects a connection ID using an implementation-specific (and perhaps deployment-specific) method that allows identification of received packets with that connection ID by the endpoint upon receipt to the appropriate QUIC connection instance.

After an endpoint receives a packet with the same connection ID and a different IP address or UDP port, it will verify the peer's ownership of the new address by sending a *challenge frame* containing random data to this new address and waiting for an echoed response with the same data. This challenge and response exchange is performed within the established crypto state, so it is intentionally challenging for an eavesdropper to hijack a session in this way. The two endpoints can continue to exchange data after the verification of the new address.

This verification is particularly useful in terms of negotiating various forms of *Network Address Translation* (NAT) behaviour. NATs are intentionally transport-aware and for TCP, NATs will attempt to maintain a translation state until it observes the closing FIN protocol exchange. UDP offers no such externally visible clues as to the ending of a session, and NATs are prone to interpreting a silent period as a signal to tear down the NAT state.

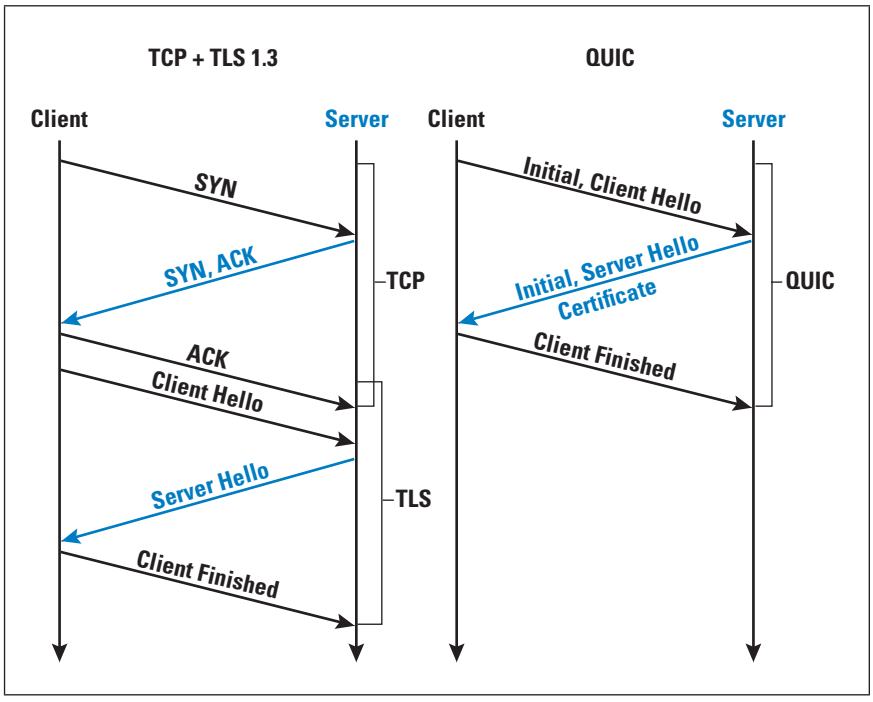
In such a case the next outbound packet might be assigned a new source address and/or UDP source port number by the NAT. It is also useful in terms of session resumption where the connection may have been idle for an extended period, and the NAT binding may have timed out. With TCP, any change in any of the four address and port fields of the connection 5-tuple will cause rejection of the packet as part of the TCP session. QUIC's use of a persistent connection ID permits the receiver to associate the new sender's address details with an existing connection.

You also can use this QUIC functionality of address agility in the context of network-level changes, such as a device switching between WiFi and cellular services while maintaining an active QUIC transport session.

QUIC Connection Handshake

A QUIC connection starts with a handshake that establishes a shared communications state and a shared secret using the QUIC-TLS protocol cryptographic handshake protocol in a single exchange. This protocol merges the TCP 3-way handshake and the TLS 1.3 3-way handshake into a single 3-packet exchange (Figure 2). This merge eliminates a full *Round Trip Time* (RTT) in the QUIC startup phase, which for short sessions is a very significant improvement in session performance.

Figure 2: TCP/TLS and QUIC Handshakes



QUIC also allows a client to send 0-RTT encrypted application data in its first packet to the server by reusing the negotiated parameters from a previous connection and a TLS 1.3 *Pre-Shared Key* (PSK) identity issued by the server, although these 0-RTT data exchanges are not protected against replay attack.

Packets and Frames

The QUIC protocol sends *packets* along the connection. Packets are individually numbered in a 62-bit number space. There is no allowance for retransmission of a numbered packet. If data is to be retransmitted, it is done in a new packet with the next packet number in sequence. That way there is a clear distinction between the reception of an original packet and a retransmission of the data payload.

You can load multiple QUIC packets into a single UDP datagram. QUIC UDP datagrams must not be fragmented, and unless the end performs *Path Maximum Transmission Unit* (PMTU) discovery, QUIC assumes that the path can support a 1,200-byte UDP payload.

A QUIC client expands the payload of all UDP datagrams carrying Initial packets to at least the smallest allowed maximum datagram size of 1,200 bytes by adding padding frames to the Initial packet or by coalescing a set of Initial packets. The payload of all UDP datagrams carrying ACK-eliciting Initial packets is padded to at least the smallest allowed maximum datagram size of 1,200 bytes. Sending UDP datagrams of this size ensures that the network path supports a reasonable PMTU in both directions. Additionally, a client that expands Initial packets helps reduce the order of amplitude gain of amplification attacks caused by server responses toward an unverified client address.

QUIC packets are encrypted individually so that the decryption process does not result in data decryption waiting for partially delivered packets. This encryption is not generally possible under TCP, where the encryption records are in a byte stream and the protocol stack is unaware of higher-layer boundaries within this stream. The additional inference from this per-packet encryption is that it's a requirement that QUIC IP packets are not fragmented. QUIC implementations typically use a conservative choice in the maximum packet size so that IP packet fragmentation does not occur.

A QUIC receiver ACKs the highest packet number received so far, together with a listing of all received contiguous packet number blocks of lower-numbered packets if there are gaps in the received packet sequence. Because QUIC uses purpose-defined ACK frames, QUIC can code up to 256 such number ranges in a single frame, whereas TCP *Selective Acknowledgment* (SACK) has a limit of 3 such sequence number ranges. This limit allows QUIC to provide a more detailed view of packet loss and reordering, leading to higher resiliency against packet losses and more efficient recovery. Lost packets are not retransmitted. Data recovery is performed in the context of each QUIC stream.

QUIC Streams

A QUIC connection is further broken into *streams*. Each QUIC stream provides an ordered byte-stream abstraction to an application similar in nature to a TCP byte stream. QUIC allows for an arbitrary number of concurrent streams to operate over a connection. Applications may indicate the relative priority of streams.

Because the connection has already performed the end-to-end association and established the encryption context, you can establish streams with minimal overhead. A single stream frame can open, pass data, and close down within a single packet, or it can exist for the entire lifetime of the connection.

By comparison, it is possible to multiplex a TCP session into streams, but all such multiplexed TCP streams share a single flow-control state. If the TCP receiver advertises a zero-sized window to the sender, then all multiplexed streams are blocked in a TCP scenario.

Each QUIC stream is identified by a unique *stream ID*, where its two least significant bits are used to identify which endpoint initiated the stream and whether the stream is bidirectional or unidirectional. The byte stream is segmented to data frames, and the stream frame offset is equivalent to the TCP sequence number, used for data-frame delivery ordering and loss detection and retransmission for reliable data delivery.

QUIC endpoints can decide how to allocate bandwidth between different streams, and how to prioritize transmission of different stream frames based on information from the application. This feature ensures effective loss recovery, congestion control, and flow-control operations, which can significantly impact application performance.

QUIC Datagrams

In addition to reliable streams, QUIC also supports an unreliable but secured data-delivery service with DATAGRAM frames, which will not be retransmitted upon loss detection[5]. When an application sends a datagram over a QUIC connection, QUIC will generate a DATAGRAM frame and send it in the first available packet. When a QUIC endpoint receives a valid DATAGRAM frame, it is expected that it would deliver the data to the application immediately. These DATAGRAM frames are not associated with any stream.

If a received packet contains only DATAGRAM frames, then the ACK frame can be delayed, as the sender will not retransmit a frame when there is an ACK failure in any case. This service is not a reliable datagram service. If a sender detects that a packet containing a specific DATAGRAM frame might have been lost, the implementation may notify the application that it believes the datagram was lost. Similarly, if a packet containing a DATAGRAM frame is acknowledged, the implementation may notify the sender application that the datagram was successfully transmitted and received.

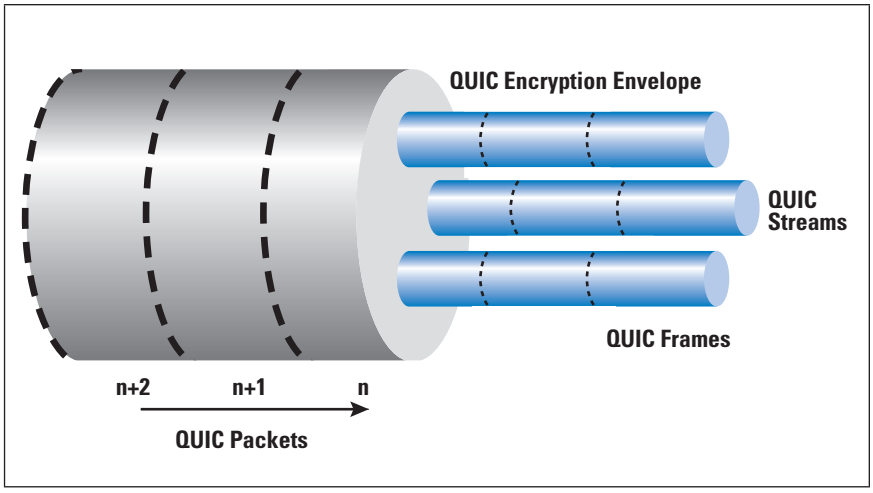
QUIC Frames

Each packet contains a sequence of *frames*. Frames have a frame-type field and type-dependant data. The QUIC standard[4] defines 20 different frame types. They serve an analogous purpose to the TCP *flags*, carrying a control signal about the state of streams and the connection itself.

Frame types include padding, ping (or keepalive), ACK frames for received packet numbers, which themselves contain *Explicit Congestion Notification* (ECN) counts as well as ACK ranges, as well as stream data frames and datagram frames.

Figure 3 shows the larger organisation of QUIC connections, streams, and frames.

Figure 3: QUIC Logical Organisation



QUIC Recovery and Flow Control

QUIC packets contain one or more frames. QUIC performs loss detection based on these packets, not on individual frames. For each packet that the receiver acknowledges, all frames carried in that packet are considered received. The packet is considered lost if that packet is unacknowledged when a later sent packet has been acknowledged, and when a loss threshold is met.

QUIC uses two thresholds to determine loss. The first is a *Packet Reordering Threshold* t. When packet x is acknowledged, then all unacknowledged packets with a number less than $x - t$ are considered lost. The second is related to the QUIC-measured RTT interval, the *waiting time* w which is determined as a weight factor applied to the current estimated RTT interval. If the time of the most recent acknowledgement is t, then all unacknowledged packets sent before time $t - w$ will be considered lost.

For recovery, all frames in lost packets where the associated stream requires retransmission will be placed into new packets for retransmission. The lost packet itself is not retransmitted.

As with TCP's advertised receiver window, QUIC contains a mechanism to enable a QUIC receiver to control the maximum amount of data that a sender can send on an individual stream, and the maximum amount on all streams at any time. Also, as with TCP, QUIC does not specify the flow-control algorithm to be used by reliable streams, although one such sender-side congestion controller is defined in [6]. This algorithm is similar to TCP's *New Reno*[7].

We now examine some problems with QUIC.

RPC Support

IP hosts commonly support just two transport services, UDP and TCP. UDP is a simple datagram-delivery service. Data encapsulated using UDP have no assured delivery. TCP, as we have seen, is a reliable streaming service. The TCP protocol repairs any packet loss or changes to the delivered packet sequence.

Another model, namely the *Remote Procedure Call* (RPC) model, emulates the functionality of procedure calls, and rather than the byte-stream model of TCP or the datagram model of UDP, the RPC model is a reliable request/reply model, where the reply is uniquely associated with the request. Perhaps the most well-known example today of an RPC framework is *gRPC*[8]. gRPC is based on an HTTP/2 platform, implying that the framework is susceptible to head-of-line blocking as with any other TCP-based substrate.

The issue here is that a reliable byte stream is not the right abstraction for RPC, as the core of RPC is a request/reply paradigm, which is more aligned to a reliable messaging paradigm, with all that such a paradigm entails. A capable RPC framework needs to handle lost, mis-ordered, and duplicated messages, with an identifier space that can match requests and responses. The underlying message transport needs to handle messages of arbitrary size, which entails packetization adaptation within the transport.

The bidirectional stream framework is a reasonable match to the RPC communications model where each RPC can be matched against an individual stream. The stream is reliable and sequenced. The data framing is not contained in QUIC, and it is still an application task to add a record structure to an RPC stream, if that is what is required. The invocation overhead is low in that the encrypted end-to-end connection is already established.

It certainly appears that HTTPS behaves much more like RPC than a reliable byte stream. That can benefit applications that run over HTTP(S), such as gRPC, and a set of *Representational State Transfer* (REST)ful APIs.

Load-Balancing QUIC

In today's world of managing scale, it is very common to place a front-end load balancer across many servers. The load balancer in the TCP world typically categorizes packets as being in the same TCP session because of a common 5-tuple value of protocol, IP addresses, and port numbers, with the confident assurance that this value is stable for the life of the TCP session.

QUIC offers no such assurances. The 5-tuple load-balancing approach can work, but if the client is behind a NAT that performs what could be called "aggressive" rebinding, then any such load-balancing approach will be thrown. The reason why is that UDP does not provide session signalling to a NAT, so there is no a priori assurance that the NAT bindings (and the presented source address and port) will remain constant for the entire QUIC session.

Now in theory IPv6 could invoke the *Flow-ID* to provide a proxy persistent field that remains constant for a flow, but the Flow-ID is of limited size and has no assurances of uniqueness, as well as evidence of highly variable treatment by IPv6 network infrastructure and end hosts.

This topic touches upon a major assumption in today's high-capacity server infrastructure on the public Internet. Data streams use TCP and the DNS uses UDP. Using UDP to carry sustained high-volume streams may not match the internal optimisations used in server content-delivery networks.

DDoS Defence

The next issue here is exposure to *Distributed Denial-of-Service* (DDoS) attacks. An attacker can send a large volume of packets to the server and cause the server to perform work to attempt to decrypt the packet. For this capability to be successful in TLS over TCP the attacker must make a reasonable guess of the TCP sequence number and window size for the packet to be accepted and passed to the TLS decoder. QUIC has no lightweight packet filter before the decoder is invoked.

On the other hand, the session encryption uses symmetric crypto algorithms, which are less of a load on the receiver to decode than asymmetric encryption. Is this difference enough to allow large-scale QUIC platforms that are DDoS resistant to be constructed? I'm unsure if there are clear answers here, but it seems that it's part of the cost of having a more complete encryption framework, which in itself appears to be sorely needed on the public Internet.

Private QUIC

For private contexts, can QUIC negotiate a "null" TLS encryption algorithm? There is a bigger world out there beyond the public Internet, and in many private data centre environments the overheads of encrypting and decrypting packets may appear to be unnecessary. While QUIC can present some clear advantages in terms of suitability to complex application behaviours in the data centre that can leverage QUIC's multi-stream capability, the cost of encryption may be too high.

Of course, there is nothing stopping an implementation using a null encryption algorithm, but such an implementation could talk only to other implementations of itself. Strictly speaking, if you remove encryption, then it's no longer QUIC and it won't interoperate with anything else that is QUIC.

QUIC and OpenSSL

It useful to ask that if QUIC has such clear advantages over TCP, then why hasn't the adoption of QUIC been rapid? Metrics of QUIC use tend to point to a use rate of some 30% of web sessions (such as Cloudflare's Radar report[9]).

However, if you alter the measurement to measure the extent to which browsers on end systems are capable of supporting a QUIC session, then the measurement jumps to 60%[10].

There are a couple of reasons why QUIC use is far lower than QUIC capability. The first is that the Chrome browser still relies on the content-level switch to QUIC, so the client has to visit the site for the first time using HTTP/2 (TCP/TLS) and thereby receive an indication if the server can support QUIC, and then on the second visit the client may use QUIC. It's not quite as simple as this, as HTTP/2 uses persistent connection, so if the second visit is sufficiently close in time to the first, then the HTTP/2 session will remain open and still be used. The Safari browser is capable of using QUIC on first use because it is triggered by the *Service Binding* (SVCB) record in the DNS, but the market share of Safari is relatively small in comparison to QUIC.

The second reason lies in the web server environment. Many servers rely on the *OpenSSL* TLS library[11], and so far, (November 2022) OpenSSL does not include support for QUIC. QUIC is supported in *BoringSSL*[12], but as the notes for BoringSSL state, BoringSSL is a fork of OpenSSL that is designed to meet Google's needs, and while it works for Google, it may not work for everyone else. Google does not recommend that third parties depend on BoringSSL. There is also *QuicTLS*, a fork of OpenSSL that Akamai and Google support[13]. This fragmentation of OpenSSL is not exactly helpful, and the result is that many server environments are waiting for OpenSSL to incorporate a QUIC library. This effort was delayed by the work on OpenSSL release 3.0.0, and then the OpenSSL folks announced their intention to provide a fully functional QUIC implementation, and this development of a new QUIC protocol stack may further delay QUIC support in OpenSSL by months, if not years. This impediment may well be the major one behind the very large-scale deployment of QUIC in the guise of HTTP/3 across the Internet.

Conclusions

We can draw a few conclusions from this effort with QUIC:

Any useful public communications medium needs to safeguard the privacy and integrity of the communications that it carries. The time when open protocols represented an acceptable compromise between efficiency, speed, and privacy are over, and these days all network transactions in the public Internet need to be protected by adequate encryption. The QUIC model of wrapping a set of transactions between a client and a server in a single encryption state represents a sensible design decision.

Encryption is no longer an expensive luxury, but a required component for all transactions over the public Internet. The added imposition is that adding encryption into a network transaction should impose no additional performance penalty in terms of speed and responsiveness.

Network transactions come in many forms, and TCP and UDP tend to represent two ends of a relatively broad spectrum. UDP is just too susceptible to abuse, so we've heaped everything onto TCP. The problem is TCP was designed as an efficient single streaming protocol, and retro-fitting multiple sessions, short transactions, shared congestion state, and shared encryption state have proved to be extremely challenging.

Applications are now dominant in the Internet ecosystem, while platforms and networks are being commoditised. We are seeing users losing patience with platforms that provide common transport services for the application that they host, and a new model where the application comes with its own transport service. This model is not just the HTTP client/server model; it has been extended into application-specific *Domain Name System* (DNS) name resolution with DNS over HTTPS. Its highly likely that this trend will continue for the moment.

Taking an even broader perspective, the context of the Internet's success lies in shifting the responsibility for providing service from the network to the end system. This shift allowed us to make more efficient use of the common network substrate and push the cost of this packetization of network transactions over to end systems. It shifted the innovation role from the large and lumbering telco operators into the more nimble world of platform software. The success of Microsoft with its Windows product was not an accident by any means. QUIC takes this success one step further, and pushes the innovation role from platforms to applications, just at the time when platforms are declining in relative importance within the ecosystem. From such a perspective, the emergence of an application-centric transport model that provides faster services and a larger repertoire of transport models, the encompassing of comprehensive encryption was an inevitable development.

References and Further Reading

[0] Geoff Huston, "A Quick Look at QUIC," *The Internet Protocol Journal*, Volume 22, No. 1, March 2019.

[1] Wesley Eddy, Ed., "Transmission Control Protocol (TCP)," RFC 9293, August 2022.

[2] Jon Postel, Ed., "Transmission Control Protocol," RFC 793, September 1981.

[3] Eric Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, August 2018.

[4] Jana Iyengar, Ed., and Martin Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021.

[5] Tommy Pauly, Eric Kinnear, and David Schinazi, "An Unreliable Datagram Extension to QUIC," RFC 9221, March 2022.

[6] Jana Iyengar, Ed., and Ian Swett, Ed., "QUIC Loss Detection and Congestion Control," RFC 9002, May 2021.

[7] Tom Henderson, Sally Floyd, Andrei Gurtov, and Yoshifumi Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, April 2012.

[8] gPRC – A cross-platform open source RPC framework: **https://grpc.io/**

[9] Cloudflare Radar: **https://radar.cloudflare.com/**

[10] APNIC Labs, QUIC Usage Report: **https://stsats.labs.apmnic.net/quic**

[11] OpenSSL a library for secure communication: **https://openssl.com**

[12] BoringSSL, an open source fork of the OpenSSL library operated by Google for internal use: **https://boringssl.googlesource.com/boringssl**

[13] QuicTLS, an open source fork of the OpenSSL library developed by Akamai and Microsoft as an interim Quic API: **https://github.com/quictls/openssl**

GEOFF HUSTON, B.Sc., M.Sc. A.M., 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**

Minimized DNS Resolution: Into the Penumbra

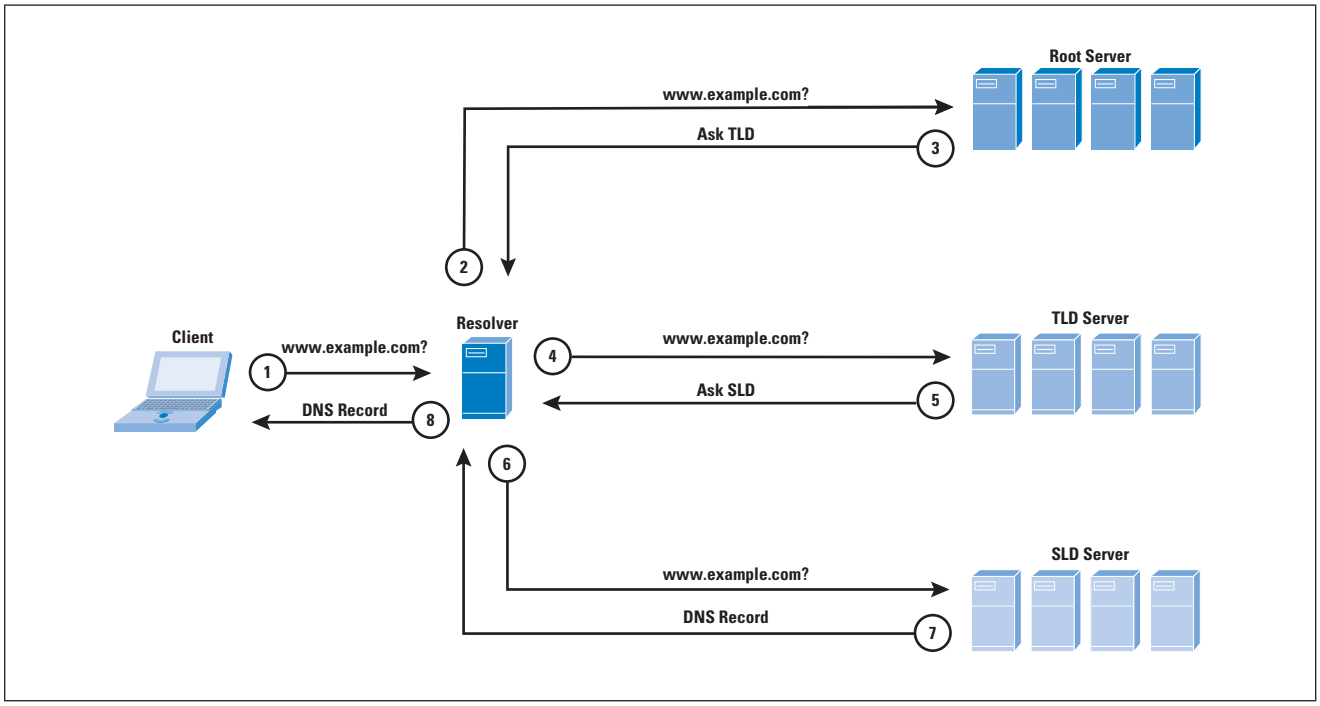
by Burton S. Kaliski Jr., Verisign

The *Domain Name System* (DNS) has long followed a traditional approach of answering queries, where resolvers send a query with the same fully qualified domain name to each name server in a chain of referrals, and, generally, apply the final answer they receive only to the domain name that was queried for. Motivated by interest in reducing both the quantity and sensitivity of information exchanged between DNS ecosystem components, DNS operators are now starting to deploy various minimization techniques that either put less information into queries or take more information out of answers, thereby reducing the need for additional queries. This article reviews four minimization techniques documented by the *Internet Engineering Task Force* (IETF), reports on their implementation status, and discusses the effects of their adoption on DNS measurement research.

DNS resolution begins with the usual occurrence that happens millions of times a second around the world: a client sends a DNS recursive resolver a query like "What is `www.example.com`'s *Internet Protocol* (IP) address?" The resolver answers, "`www.example.com`'s IP address is `93.184.216.34`."

Many clients may use the same resolver, so the resolver may already have a response to the query in its cache. If the resolver has an empty cache, it will interact with the authoritative name-server system using a protocol flow such as shown in Figure 1.

Figure 1:Textbook DNS Resolution



1. The client asks the resolver, "What is **www.example.com**'s IP address?"

2. The resolver queries one of the DNS's 13 root servers[1] for an answer to question 1.

3. The root server responds with a referral-type response directing the resolver to the name server for the *Top-Level Domain* (TLD) in the query name, that is, the **.com** name server.

4. The resolver sends the query to the TLD server.

5. The TLD server refers the resolver to the name server for the *Second-Level Domain* (SLD), that is, the **example.com** name server.

6. The resolver sends the query to the SLD server.

7. The SLD server returns one or more DNS records that specify **www.example.com**'s IP address.

8. The resolver relays the DNS records to the client.

The referrals in steps 3 and 5 are a result of the delegation structure of DNS. The root zone has delegated the authority for responding to queries for domain names within existing TLDs to TLD servers. Many TLD zones have similarly delegated the authority for responding to queries within SLDs to SLD servers. In step 7, the SLD server has the authority to respond for the domain name **www.example.com**.

The DNS standard (based on RFC 1035[2] and other documents)—as well as current practice—include many more details. For purposes of this article, the "textbook DNS" described here is an effective starting point, but two additional details may be helpful in framing the techniques that follow:

- If a name server knows that a domain name doesn't exist, then it returns the negative response code (rcode 3), typically referred to as NXDOMAIN. (Otherwise, either the domain name exists and the name server is authoritative for it and returns a positive answer along with rcode 0; or the name server is not authoritative and returns a *referral*.)
- If a resolver and a name server implement the *Domain Name System Security Extensions* (DNSSEC)[3], the resolver asks the name server to include DNSSEC information in its response, and if the domain name doesn't exist, then the name server also returns an NSEC[4] or NSEC3[5] record that specifies two endpoints between which no other domain names exist, for some ordering of domain names. With NSEC, the ordering is based on the domain names themselves; with NSEC3, it's based on their hash values. Either way, the resolver receives information demonstrating not only that the queried name doesn't exist, but also that *other* domain names between the endpoints don't exist.

(The records are formed this way so that they can be precomputed and signed when the name server is provisioned, based on domain names that do exist in a zone. The name server then already has the information it needs to respond to a query for a nonexistent domain name, without having to sign responses in real time, although some name servers do support dynamic signing.)

It's clear from a brief review of Figure 1 that textbook DNS resolution includes more information in DNS exchanges than necessary. This fact is particularly evident on the resolver-to-root exchange, where the resolver queries for a fully qualified domain name, yet the root server responds with a referral involving just the TLD. But the observation holds at other levels as well.

Forwarding fully qualified domain names may have historically simplified implementation, in that the resolver either gets the answer to a query from its cache, or it forwards the same query to a succession of name servers. This practice also minimizes the depth of the iterative resolution process, because the query includes enough information for each name server either to refer the resolver to another name server, or to answer the query itself (if the query wasn't fully qualified, then a name server might respond with a referral to itself in some cases, an unnecessary extra step). However, the textbook approach doesn't leverage all information available to the resolver, either from DNS or from other sources. Indeed, a fully qualified domain name, while convenient from an implementation perspective, may include more information than the name server needs to know.[6]

Minimized DNS Resolution

Minimized DNS resolution encompasses an emerging set of techniques that bring the resolver-to-authoritative traffic closer to the need-to-know principle, while still facilitating DNS resolution. Four such techniques have received the most attention, each reducing the quantity and/or sensitivity of information exchanged between resolvers and authoritative name servers in a different way. Documented by the IETF's *DNS Operations* (DNSOP) working group, the techniques include:

- Query Name (or qname) Minimization, described in RFC 9156[7];
- NXDOMAIN Cut Processing, described in RFC 8020[8];
- Aggressive DNSSEC Caching, described in RFC 8198[9]; and
- Local Root (sometimes called "hyperlocal") and other locally served zones, described (in the case of the root zone) in RFC 8806[10].

Important from an operational perspective, all four can generally be applied by a resolver on its own, without any coordinated changes by authoritative name servers, other than the participating name server conforming with previous DNS specifications. (The locally served zones technique requires that the zone data be made available.)

RFC 8932, produced by the IETF's *DNS Private Exchange* (DPRIVE) working group, encourages implementation of all four techniques to reduce both the quantity and sensitivity of "data sent onwards from the [recursive resolver] server"[11]. (DPRIVE and other IETF working groups have also developed specifications for DNS encryption, but they are outside the scope of this article.)

The techniques can generally be adopted for interactions between resolvers and authoritative name servers for any zone. (They don't apply to the client-resolver exchange.) They are particularly beneficial for interactions with the root and TLD servers, for at least two reasons:

1. The primary purpose of the root and TLD servers is global navigational availability: referring requesters to other name servers that are actually authoritative for a response. A fully qualified domain name (or even a full set of queries) is therefore not generally needed at these servers, only enough information to make the referral, making minimization techniques appropriate options. But high-availability service is paramount, favoring techniques with low operational risk.

2. Because of the recursive, cached architecture of DNS, the sensitivity of the traffic on these exchanges is already relatively low compared to other parts of the DNS ecosystem, such as the client-to-resolver exchange. In particular, because the resolver is between the client and the authoritative name servers, its queries to the authoritative name server conceal the client's identity and instead represent aggregate interests of clients. (Moreover, although information about the client's IP address may be conveyed in a query via the "client subnet" option[12], it is specifically recommended that this extension not be included in queries to the root and TLD servers.) Minimization techniques can therefore arguably lower the sensitivity of the information on the resolver-to-root and -TLD exchanges sufficiently that techniques with higher operational risk such as DNS encryption become questionable from a cost-benefit perspective, compared to disclosure risks on other exchanges such as client-to-resolver[13].

Minimization techniques also can improve resolver performance, given that they enable a resolver to answer more queries on its own, and thereby respond more quickly. They can likewise improve performance for name servers, which will receive less unnecessary traffic—including attack traffic that might have leveraged a resolver as an intermediary. And as minimized traffic becomes the "new normal" on these exchanges, it may become easier for name servers to detect and deflect other types of attack traffic, which will become more "abnormal."

Even if a resolver implements DNS encryption, it still makes sense for the resolver to implement minimization techniques to reduce the amount of information disclosed to name-server operators.

Minimization opens a new chapter in DNS resolution. With the new techniques, the traditional DNS resolution process is updated with a new approach optimized for the global DNS as it exists today, balancing confidentiality and availability objectives. The first minimization technique is perhaps the most fundamental, as it changes the most apparent nonminimized feature of textbook DNS: sending the fully qualified domain name to each name server in the chain of referrals. Note: In 2015, Verisign announced a royalty-free license to its qname minimization patents in connection with certain IETF standardization efforts. For more information, refer to IETF IPR disclosure 5197.

Query Name (Qname) Minimization

It is just a "tradition" that resolvers send the fully qualified domain name at each level of the DNS hierarchy, not a requirement of the DNS specifications. In the words of RFC 9156 (first reported by Stéphane Bortzmeyer in RFC 7816[14]), the tradition is motivated by an early goal of minimizing the number of queries that might need to be made:

> In a conversation with the author in January 2015, Paul Mockapetris explained that this tradition comes from a desire to optimise the number of requests, when the same name server is authoritative for many zones in a given name (something that was more common in the old days, where the same name servers served `.com` and the root) or when the same name server is both recursive and authoritative (something that is strongly discouraged now).

This practice, as discussed previously, can also optimize the number of requests when a name server is authoritative for only one zone.

The consequence of the tradition is that the resolver included more information than necessary in each query. Although the risk of disclosure of sensitive information on the resolver-to-root and -TLD exchanges is relatively low, as discussed previously, it would be better, per the principle of minimum disclosure, to send only as many labels as the name server needs to make a referral. Any labels beyond that point are extraneous information.

One way to reduce the amount of information disclosed is to remove one or more of the extraneous labels. In this "omitted-label" approach to reducing the amount of information included in a query to an authoritative name server, the query name `www.example.com` in the request to the root server could be replaced simply with the TLD, that is, with `.com`.

Another way is to replace one or more of the extraneous labels with random or other alternative labels. As examples of a "false-label" approach, we could replace `www.example.com` with $\texttt{<r}_3\texttt{>.<r}_2\texttt{>.com}$ or with $\texttt{<r}_2\texttt{>.com}$, where $\texttt{<r}_2\texttt{>}$ and $\texttt{<r}_3\texttt{>}$ are randomly generated labels. Another real-world qname minimization technique suggested replaces `www.example.com` with `_example.com`.[7]

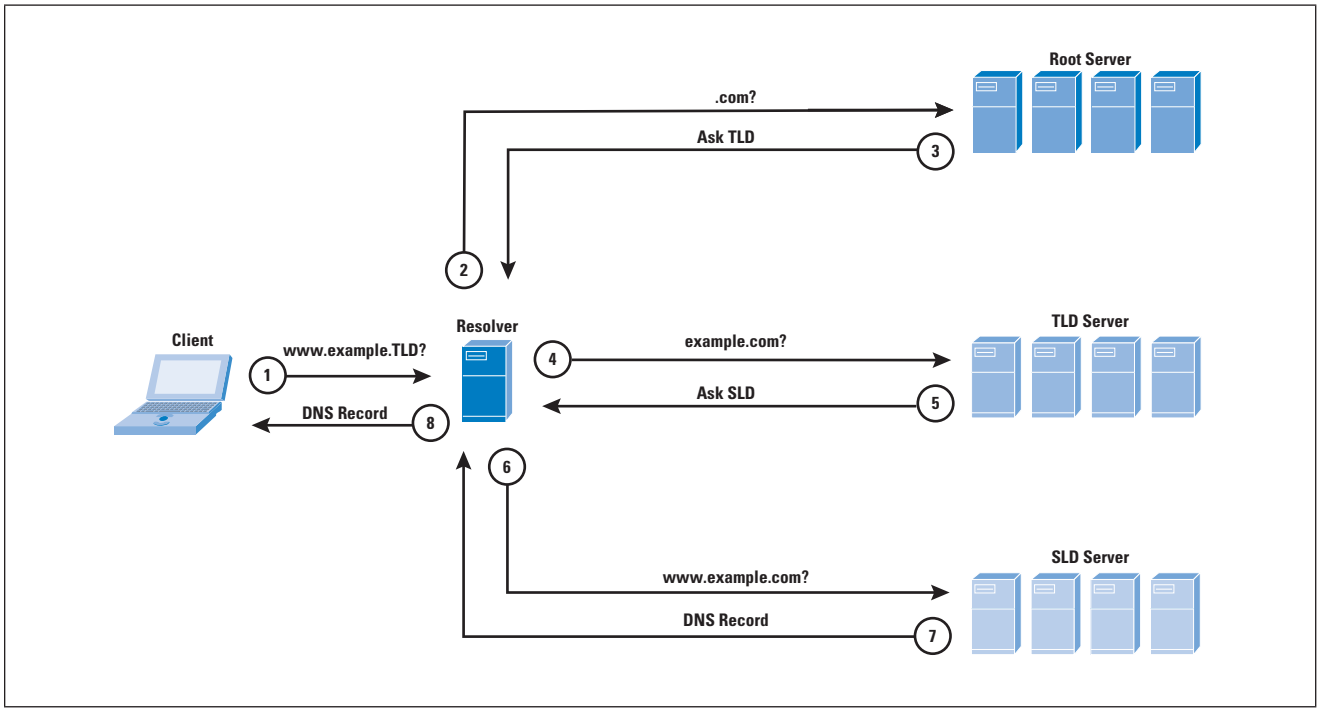
The RFC adopts the omitted-label approach for *query name* (or qname) minimization (or as it is spelled in the RFC, "minimisation").

A resolver implementing qname minimization, as described in RFC 9156, takes advantage of information about how the DNS hierarchy is organized today at its higher, navigational levels, such as the root server delegating authority for existing TLDs to TLD servers, and typical TLD servers delegating authority for existing SLDs to SLD servers. As shown in Figure 2, when the resolver queries the root server as part of resolving a domain name, it sends only the TLD label to the root server. When it queries the TLD server, it sends only the SLD and TLD labels. It can also take advantage of the knowledge that some TLD servers delegate authority for some of their hierarchy at the third level rather than the second level, as discussed previously, thereby saving a step in those cases. The *Public Suffix List* (PSL)[15] is a potential source for information about where these delegations or zone cuts may occur, as Geoff Huston has observed[16].

In addition to replacing or removing labels, the resolver can also change the *Query Type* (QTYPE) from the one the client requested, to further reduce the amount of information disclosed. RFC 9156 recommends setting the QTYPE to "A" or "AAAA" regardless of the actual record type of interest, except for the final query with the full query name.

A resolver can apply qname minimization to its interactions with any authoritative name server at any level of the DNS hierarchy, and the name server won't have to do anything differently. The name server will just receive queries with less information in them, except for the final name server in the chain.

Figure 2: DNS Resolution with qname Minimization



Qname minimization therefore provides a valuable information protection tool for both resolver operators and their users. Indeed, as Basileal Imana, Aleksandra Korolova, and John Heidemann state in their study of institutional privacy risks, "The currently available best way for institutions to reduce information leakage is to run their own resolver, and deploy query name minimization".[17]

Resolver operators have encountered one complication in deploying qname minimization: the *Empty Non-Terminal* (ENT) *Problem*, as described in RFC 7816[14]. The problem can cause a resolver to continue to send queries during the minimized iterative resolution process, even after it should have become clear that the fully qualified domain name doesn't exist. Retrying with the fully qualified domain name in the presence of ENTs wouldn't disclose more information than the resolver would have been sent with traditional DNS resolution, but it would generate unnecessary additional queries. While it has become common practice simply to stop qname minimization after three labels, the underlying ENT problem remains. Resolving this complication is the focus of the next technique.

NXDOMAIN Cut Processing

NXDOMAIN, the negative answer in DNS, technically means that a domain name doesn't exist—and therefore, by definition, that it has no subdomains.

However, because of the ENT ambiguity just mentioned, resolvers have traditionally limited their interpretation of NXDOMAIN to the domain name itself. This tradition has resulted in both additional workload for the resolver and extra traffic to the name-server system.

NXDOMAIN cut processing, described in RFC 8020[8], expands the interpretation. As the title of the RFC states, a resolver implementing this technique interprets NXDOMAIN as "there really is nothing underneath;" the DNS tree is "cut." In support, the RFC, authored by Stéphane Bortzmeyer and Shumon Huque, updates the DNS specifications to state that a name server must return NODATA in response to a query for an ENT, thereby resolving the ENT ambiguity.

Similar to qname minimization, a resolver can apply NXDOMAIN cut processing to its interactions with any authoritative name server. The name server doesn't have to do anything differently as long as it handles ENT queries correctly. It will just receive less traffic.

With the root zone not having any ENTs, and with careful consideration given to the risks of ENTs in TLD zones[18], it's reasonable for resolvers to implement NXDOMAIN cut processing at the root and TLD levels of the DNS hierarchy, consistent with the deployment of qname minimization at those levels. Processing for additional zones can be enabled as resolver operators gain more confidence in the corresponding name servers' handling of ENTs.

Or resolvers could simply adopt the technique unilaterally, regardless of the name server's behavior, a decision endorsed by RFC 8020:

> "Such name servers are definitely wrong and have always been. Their behaviour is incompatible with DNSSEC. Given the advantages of 'NXDOMAIN cut,' there is little reason to support this behavior."

NXDOMAIN cut processing helps qname minimization by enabling a resolver to stop the minimized iterative resolution process as soon as it receives an NXDOMAIN answer, meaning that the resolver will disclose less information in its traffic when a domain name doesn't exist, just as it discloses less when a domain name does exist. The combination of the two techniques can also be effective in defending against certain attacks, such as random subdomain attacks, where an adversary queries one or more resolvers for random subdomains of a common ancestor. With traditional processing, the resolvers will forward the subdomain queries to the ancestor's name server, generating a volumetric attack where the name server can't see the attack's original source. If the ancestor exists and is protected by DNSSEC, then aggressive DNSSEC caching can help a resolver reduce the number of additional subdomain queries that it forwards, as described by Petr Špaček.[19] If the ancestor doesn't exist, then NXDOMAIN cut processing can keep the resolver from forwarding further subdomain queries after it knows that the ancestor doesn't exist.

With NXDOMAIN cut processing, a resolver broadens its interpretation of a negative answer to draw conclusions about subdomains of a domain name that it previously queried for. The next technique does something similar for negative answers in the DNSSEC case, drawing conclusions about other domain names in the zone as well.

Aggressive DNSSEC Caching

As discussed previously, negative answers in DNSSEC—in the form of NSEC and NSEC3 records—indicate that no domain names exist between two endpoints in some ordering of domain names. (One further detail: with the opt-out flag set in NSEC3, some domain names between the endpoints may actually exist, but not have DNSSEC-signed delegations. If a resolver is interested only in domain names that can be validated with DNSSEC, then the NSEC3 record is still useful information.)

Resolvers traditionally haven't taken advantage of the information these records provided about the nonexistence of other names, however.

Even though NSEC and NSEC3 records provide enough information for a resolver to conclude on its own that other domain names between the endpoints don't exist, resolvers have traditionally limited their interpretation to the domain name that was queried for.

Although authoritative name servers return NSEC or NSEC3 records in response to queries for both nonexistent domain names and ENTs, it's possible to tell the two classes apart, as detailed in Appendix B of RFC 8198[9] for NSEC and in Sections 8.4–8.8 of RFC 5155[5] for NSEC3.

The narrow interpretation is actually the correct one according to the original DNS specifications, not the result of an ambiguity as it was for the previous technique. RFC 4035[3] describes the limitation as a "prudent" approach:

> "In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace."

The limitation may once again result in the resolver doing more processing and sending more queries than it needs to, given the information it already has on hand.

Aggressive DNSSEC caching, described in RFC 8198[9], takes a broader interpretation. The RFC, authored by Kazunori Fujiwara, Akira Kato, and Warren Kumari, updates the DNS specifications to state that a resolver may handle client queries for domain names that fall between the endpoints of previously received NSEC and NSEC3 records on its own. (It also allows the resolver to apply wildcard records to names between the endpoints when matching wildcard records exist.)

The technique offers an excellent illustration of the relative nature of the minimum disclosure principle, and it also improves the resolver's protection against random subdomain attacks as detailed in the previous section. Without DNSSEC, a resolver would need a name server's help for each new domain name it processes that's not a subdomain of a nonexistent domain. With DNSSEC, the resolver no longer needs as much help, so the threshold for minimum disclosure is reduced.

Similar to the two previous techniques, a resolver can apply aggressive DNSSEC caching to its interactions with any name server at any level. The name server again doesn't have a direct operational role and will just receive less traffic. The name server must handle NSEC or NSEC3 correctly, which is less of a concern than for NXDOMAIN and ENTs, given that the DNSSEC accounts for ENs.

The foregoing has three caveats:

First, as mentioned already, if an NSEC3 record has an opt-out flag, the resolver can't conclude that other domain names between the endpoints don't exist, only that they don't have secure delegations.

It therefore can't apply aggressive DNSSEC caching to such a record. Given that NSEC3 is the predominant choice for TLDs, and that the opt-out flag is commonly used[20], aggressive DNSSEC caching will generally not help at the TLD level.

Second, the reduction in the number of queries sent assumes that the NSEC or NSEC3 endpoints actually span multiple domain names. Both techniques have variants, documented in RFC 4470[20] and RFC 7129[21], where the returned endpoints effectively span only the one domain name of interest, taking away the advantage of aggressive DNSSEC caching. Moreover, some implementations of these variants incorrectly report that some resource record types don't exist, possibly resulting in a resource record becoming unresolvable.[22, 23] The "aggressive" interpretation of negative DNSSEC responses makes implementation errors more consequential as well.[24]

Third, as Geoff Huston has observed,[25] "the results [of aggressive DNSSEC caching] may not be that promising" for resolvers that load-balance their queries into servers with independent caches, for example, based on a hash of the query name.

These caveats aside, if the resolver were somehow to cache every NSEC or NSEC3 record in a pre-signed zone, and if there were no NSEC3 opt-outs, and if the ranges within the records collectively spanned the entire zone, then the resolver would be able to handle queries for every nonexistent domain name in the zone on its own, for as long as the records were valid.

If the resolver likewise were to cache every existing DNS record in the zone, then it could handle queries for existing domain names too.

A resolver might be able to bring all of these records into its cache if the set of queries it sends is directed, at least in part, by a carefully designed process. Using aggressive DNSSEC caching, the resolver will cache the NSEC record as evidence that domain names between the endpoints don't exist. In addition, it can cache the record as evidence that the two domain names at the endpoints do exist. Then the resolver can simply query for the DS and NS records for the two domain names at the endpoint, and it will have obtained the full DNS records for the two endpoints from this zone file. The resolver can repeat the process with other random long domain names until it has obtained a set of NSEC records that collectively span the zone. After sending queries for the zone's own DNSKEY, NS, and SOA records, the resolver will have obtained all the DNS records in the zone file. If the resolver implements NXDOMAIN cut processing and aggressive DNSSEC caching, it will then be able to answer client queries for every domain name without making further queries to the zone's authoritative name server. This process does not work well for NSEC3 and NSEC5.[26, 27] By populating the resolver's cache in this way, the client would remove its own and other clients' interests in domain names from future resolver-to-authoritative queries.

But if the resolver just wants to avoid sending queries to a remote name server for a zone entirely, the next technique offers a more direct way to achieve the goal if the zone is appropriately configured.

Locally Served Zones

The DNS resolution processes shown in Figures 1 and 2 maintain a clear distinction between DNS ecosystem components: the client is separate from the resolver, which in turn is separate from the authoritative name servers. The separation implies a potentially global communications path between components, leading to the information disclosure concerns that have been the focus of this article.

But what if the communications path between two components was instead a local one? Such locality would not be unprecedented. Indeed, the resolver is often located within the same network as the client, which as discussed previously was one of the reasons for the relatively late standardization of an encrypted DNS protocol for the client-to-resolver exchange. An authoritative name-server instance can similarly be located within the same network as the resolver, as long as it can somehow be provisioned with a current copy of the zone file.

RFC 8806[10], authored by Warren Kumari and Paul Hoffman, describes how to run a local instance of authoritative zone data with two constraints. First, the specification is limited to the root zone. Second, the local instance must indeed be run locally: that is, it must be accessible only to the resolver, and therefore not visible to other servers on the network. (Deploying the local instance at a loopback address, as proposed in the title to RFC 7706[28], the predecessor to RFC 8806, is one way to ensure locality.)

ICANN's CTO organization describes the local root technique as "hyperlocal," and its OCTO-016 technical note[29] proposes the technique as a way to "[improve] the decentralization of the root name service to mitigate risks that the [Root Server System] may face over time."

While OCTO-016 focuses on improving decentralization, and RFC 7706, per its title, on decreasing access time, it's also clear that the locally served zones technique also reduces the amount of information disclosed on the resolver-to-authoritative exchange. Indeed, RFC 8806 states that in addition to decreasing access time (particularly for negative responses), another goal of the technique is "to prevent queries and responses from being visible on the network."

A resolver can in principle get a copy of a zone file just like an authoritative name server might, via a zone-transfer protocol such as *Authoritative Transfer* (AXFR), described in RFC 5936[30], and *Incremental Transfer* (IXFR), described in RFC 1995[31]. These protocols give options for downloading a full zone file and for obtaining incremental updates respectively and may be enabled by a name server, depending on zone policy.

An encrypted version of these protocols, called *XFR-over-TLS* (XoT), is currently in development[32]. Another alternative is for the zone data to be made available for download at a web address via the *Hypertext Transfer Protocol Secure* (HTTPS) protocol. For instance, ISI's *LocalRoot* project[33] provides access to copies of the root zone, as well as the **`.arpa`**, **`root-servers.net`** and **`dnssec-tools.org`** zones.

In addition, the new ZONEMD record, described in RFC 8976[34], provides a way to authenticate the integrity of a downloaded zone file (in contrast to DNSSEC, which authenticates individual sets of records).

Locally served zones and zone digests are more practical for small, slowly changing zones, such as the root zone, than for large, fast-changing ones. RFC 8976 states:

> "ZONEMD is impractical for large, dynamic zones due to the time and resources required for digest calculation."

The locally served zones technique, like others in this article, is another one that a resolver can apply to any zone at any level, in this case as long as zone data is made available for download. Its operational characteristics are similar to the other techniques: the name server doesn't need to do anything differently; the changes are all on the resolver's side (in terms of the resolution protocol); and the name server will receive less traffic (in this case, no traffic). The zone operator will need to provide a zone-transfer service, but this change is in provisioning, rather than resolution.

The technique does come with one significant caveat. The traditional DNS architecture with its resolver-to-authoritative exchanges has been optimized for the case where the operator for a zone is aware (or in the case of the root server, the multiple operators are collectively aware) of all of the name servers that are serving the zone. The operator(s) are therefore in a position where they can potentially check the consistency of the zone file information served by all these servers.

Until new mechanisms for synchronization are in place, locally served zone instances would fall outside a typical zone operator's awareness and ability to check consistency. OCTO-016 recognizes the need for additional work in stating:

> "If hyperlocal were to see a significant uptake, a new system for root zone distribution would need to be devised to satisfy the reliability and scalability requirements associated with the widespread hyperlocal deployment in recursive resolvers."

A system with these characteristics will be important if and when resolvers do adopt the locally served zones technique more broadly. But in the meantime, for resolvers that implement locally served zones, the technique will achieve the ultimate in minimum disclosure of information about client interests in domain names in the zone. The traditional resolver-to-authoritative exchange for these zones will have no conventional DNS queries at all.

Implementation Status

The minimization techniques described in the previous four sections are gradually being implemented and deployed in the DNS ecosystem. The following is a sampling of support by selected resolver operators and open source resolvers as of this writing.

A note on methodology: The distributed DNS ecosystem has tens of millions of resolvers[35]. The ones referenced here are based on the list of "major Open DNS resolvers" in Huston's qname minimization deployment study[36], plus those in Mozilla's *Trusted Recursive Resolver* (TRR) program[37]. The determination of whether a resolver supports a technique is based primarily on public announcements. However, Huston's study is also cited as likely evidence of qname minimization support. The open source resolver packages considered match the list included in Wouter de Vries et al.'s paper on qname minimization.[38]

Qname Minimization

Qname minimization is included in the BIND[39], *Knot Resolver*[40], *PowerDNS*[41], and *Unbound*[42] open source resolver software packages. Cisco *Umbrella*[43], Cloudflare's 1.1.1.1[44], Comcast's *Xfinity Internet Service*[45] (by virtue of its inclusion in Mozilla's TRR program, which requires the capability), *Google Public DNS* (as related by Moura *et al.*[46]), and *NextDNS*[47] have all announced that they have implemented qname minimization. Google Public DNS has also reported that it uses a "nonce prefixes" technique where extraneous labels are replaced with a random label, an example of the "false-label" approach mentioned previously.[48]

In addition, *dnswatch*, dyn *Recursive DNS*, *Quad9*, Neustar *UltraDNS Public*, and *Hurricane Electric* (HE) resolvers have been observed in Huston's study as likely to be supporting qname minimization. The deployment of qname minimization has also been the subject of Internet measurement studies. De Vries observed that as early as April 2017, "0.9% (82 of 9,611) of RIPE Atlas probes had at least one [qname-minimizing] resolver," and by October 2018, the percentage had grown to 11.7%.[49] As of August 2021, NLnet Labs' measurement dashboard showed that 47.8% of such probes interact with a qname-minimizing resolver.[50]

Huston reported that as of mid-2020, "some 18% of users pass their queries through resolvers that actively work to minimize the extent of leakage of superfluous information in DNS queries," adding that the percentage had increased from 3% since a year prior. Huston later clarified that the percentages likely underestimate actual adoption because the study's active DNS measurement technique uses four-label client queries. Many resolver implementations of qname minimization revert to ordinary DNS resolution after three labels, potentially making the particular measurement technique undetectable by the study's test servers.[51]

In the same timeframe, according to research published by Matt Thomas,[52] nearly half of all queries received by the `.com` and `.net` TLD servers consisted of only two labels.

The comparable percentage two years prior was 30%. The increase in two-label queries was accompanied by a similar decrease in three-label queries and thus can be taken as an indicator that qname minimization is being deployed at many resolvers. The upward trend has continued, reportedly reaching 55% as of February 2021.[53] It should be noted, however, that many factors contribute to the composition of traffic to authoritative name servers, and the fraction of queries that have a certain number of labels may not be directly reflective of the fraction of resolvers that support qname minimization, nor with the fraction of users who interact with such resolvers.

NXDOMAIN Cut Processing

Knot Resolver,[54] PowerDNS,[55] and Unbound[56] all support NXDOMAIN cut processing. BIND lists the technique as supported but made obsolete by Aggressive DNSEC Caching.[57] No announcements by recursive DNS operators have been found as of this writing. However, it is likely that many do support the technique, given that, as discussed previously, NXDOMAIN cut processing is not a new feature but rather the lack of accommodation for an old bug.

Aggressive DNSSEC Caching

Aggressive DNSSEC caching is included in BIND[58], Knot Resolver[59], PowerDNS[60], and Unbound.[61, 62] Cloudflare has reported that it has implemented aggressive DNSSEC caching,[44] as well as Google.[63]

Hyperlocal Zones

BIND[64], Knot Resolver[65] (following a "pre-filling" technique that RFC 8806 reports is consistent with the RFC's requirements, but which diverges from the technique specified in RFC 7706), and Unbound[56] all support hyperlocal zones. No announcements by recursive DNS operators were found.

Impact on DNS Measurement Research

The resolver-authoritative exchange has historically given authoritative name servers at all levels of the DNS hierarchy insights into the domain names being queried by a resolver's clients. While the recursive, cached architecture of the DNS ecosystem conceals the identity of the specific client that originated a query, the receipt of a fully qualified domain name by an authoritative name server nevertheless reveals that *some* client is interested in the name. With the traditional DNS resolution process, that information potentially reaches all levels, starting with root and TLD.

One of the studies facilitated by this information was the DNS community's research into name collisions related to the introduction of new *generic TLDs* (gTLDs) to the global DNS.

Root-server traffic already had shown significant evidence that resolvers (and therefore clients) were making many queries for domain names in TLDs that were not part of the global DNS[66]. The root servers had historically, and correctly, responded that such domain names didn't exist, leading clients to query for different domain names (or to give up). But if a new TLD were added to the global DNS, the root servers (together with other servers) might begin to respond positively to client queries for domain names in the TLD. That change might then cause legacy clients to connect, inadvertently, to new, external servers—a name collision.

Because root servers had information about non-existent TLDs of interest to clients, as well as fully qualified domain names, researchers were able to determine not only which new gTLDs were already being queried for, but also which domain names within those new gTLDs were being queried. One of the sources for this information was the *Day in the Life* (DITL) exercise run annually by the *DNS Operations Analysis and Research Center* (DNS-OARC)[67]. Researchers also performed additional analysis based on their own data sources and reported findings at a workshop on name collisions[68].

The insights from root-server query data led to the identification of various network and client configurations that might be at risk if a new gTLD were delegated. For example, researchers identified vulnerabilities related to the *Web Proxy Auto-Discovery Protocol* (WPAD)[69, 70, 71]. Researchers also found an operating-system vulnerability that did not involve new gTLDs based on their review of root-server query data[72]. Verisign later conducted an outreach program that mitigated a broad range of name collision risks, again drawing from the query data[73].

It is quite possible that if the minimization techniques described in this article had been broadly adopted a decade ago, researchers would not have been as able to study name collision risks as effectively, at least based on analyzing root-server data. One of the co-discoverers of independent vulnerability, simMachines—co-discoverer of the bug—is quoted in a blog post on qname minimization[6] as stating that the "analysis would have been partially impacted" if fully qualified domain names had not been visible in root-server traffic.

The loss of visibility is exactly what should be expected, inasmuch as the goal of each of the minimization techniques is to reduce root and TLD servers' visibility into clients' interests in domain names. Adoption of the techniques will impact DNS measurement research at root and TLD servers in different ways.

- Qname minimization and NXDOMAIN cut processing, which amplify one another, reduce root and TLD servers' visibility into the lower-level domains that a resolver (and by implication, its clients) may be interested in. As more resolvers adopt qname minimization with an omitted-label approach, the overall query traffic to the root servers will trend toward single labels, while the traffic to the TLD servers will trend toward two or three labels depending on the delegation structure.

  If these techniques had been in place at a given resolver when the name collisions research was performed, the root-server data associated with this resolver would only have indicated the TLDs the resolver was interested in, not the fully qualified domain names. Potential collisions between legacy systems and new gTLDs might have been highlighted, but some of the detail that helped determine the reason for the query and the impact of a positive response may have been obscured.

- Aggressive DNSSEC caching similarly reduces root and TLD servers' visibility into a resolver's interests in non-existent domain names that happen to be between the NSEC or NSEC3 endpoints obtained in response to another recently queried non-existent domain name. If aggressive DNSSEC caching had been in place at a resolver during the name collisions research, the root-server data associated with the resolver may have provided only partial information about the non-existent TLDs the resolver and its clients were interested in. This limitation on visibility may also have made it harder to assess the degree of risk associated with a given new gTLD.

- Finally, hyperlocal zones reduce the visibility of a name server into a participating resolver's interests entirely. ICANN's CTO organization, in its technical analysis of the hyperlocal root-zone technique[74], aptly summarizes the impact on telemetry as follows: "…one likely consequence of significant hyperlocal root service deployment will be a general decrease in knowledge about how the global DNS operates."

We could make similar observations about other observations and actions motivated by root-server data. For instance, Matt Thomas' and Duane Wessels' study of DNS traffic to the root generated by Chromium-based browsers[75] depends on statistics about queries to the root servers for non-existent TLDs. While qname minimization would not affect the statistics (the queries are already a single label), aggressive DNSSEC caching might. And the "mysterious root query data"[76] reported by Duane Wessels and Christian Huitema, which includes many query names consisting of random 12- and 13-character SLDs followed by existing TLDs, would not have been seen if the resolver(s) that sent the queries had implemented qname minimization with an omitted-label approach. (To be fair, initial community feedback[77] attributes the data to a different approach to reducing the amount of information in queries to the root server: the "nonce prefixes" technique previously mentioned in connection with Google Public DNS[48].)

As minimization techniques are applied to the resolver-to-root and -TLD exchanges, researchers will need to expand their use of data sets from other parts of the DNS ecosystem—appropriately anonymized and summarized for sharing—if they want to maintain a larger view of the types of queries that clients are making. There are already numerous approaches for sharing data outside the resolver-to-root and -TLD exchanges.

DNS-OARC already collects research data from other "busy and interesting DNS servers," not just root servers. Passive DNS tools[78] offer an alternative approach for analyzing DNS query traffic patterns at an ecosystem level. And query data specific to security vulnerabilities can be shared with general threat-indicator tools.

The resolver-to-root and -TLD exchanges themselves will likely still have interesting data for researchers as well. Indeed, studies of these exchanges will provide important insights into the deployment of minimization techniques, which will be a gradual process over many years. Such studies may give even more information about the configuration of individual resolvers than was previously available when resolver behavior was more uniform.

Researchers may also be able to infer statistical information about the resolver selections of certain client environments, by measuring how known changes in these environments are filtered through the resolvers of different configurations. One potentially fruitful area for such research: the new HTTPS resource record[79]. The record is gradually being introduced with early support by Apple's iOS 14 and macOS 11 operating-system betas[80]. Clients that support the HTTPS record will typically make three queries to their resolver, for the A, AAAA, and HTTPS record types. Traditional resolvers will then forward queries of all three types to the root and TLD servers. But resolvers that implement qname minimization may send only minimized A type queries to get a referral to the server that is actually authoritative for all three. The presence of HTTPS queries on the resolver-to-root and -TLD exchanges for a given resolver will therefore be an indicator not only that the resolver likely isn't yet applying qname minimization, but also that a portion of the clients that query for the HTTPS record type are using the resolver.

Just as minimization techniques represent a new chapter in DNS protocol evolution, they also will bring a new era in DNS measurement research. DNS resolution will still be taking place, although in different ways, and data analysis will still be possible, but with alternate arrangements. Such alternatives will likely depend more on active measurement techniques where clients send queries that are designed to be detectable even if minimized resolution is taking place. Both the practice and the study of DNS will go on.

Conclusion: Into the Penumbra

For the past few decades, as DNS resolution has followed the textbook DNS approach shown in Figure 1, DNS operators have had significant visibility into aggregate client interests in domain names. While the visibility, as noted earlier, has not included information about specific client identities, it has included fully qualified domain names, forwarded to each authoritative name server in the chain of referrals.

As minimization techniques are deployed, less information will be sent on the resolver-to-authoritative exchange, especially at the root and TLD levels, both because individual queries will include less information (for example, because of qname minimization), and because fewer queries will be sent (because of the other techniques). That's a gain for the need-to-know principle, which is the primary motivation for the change. But it's also a loss for DNS measurement research—at least for the passive measurement research based on assumptions that textbook DNS is deployed.

Because DNS resolvers are gradually deploying minimization techniques, rather than adopting all at once, they are like an eclipse: a slow and steady occlusion of the information content of the resolver-to-authoritative DNS exchange. The minimization eclipse likely will never be a total one, as many legacy DNS resolvers will continue doing what they've been doing all along. But its effects will be noticeable, and, inasmuch as the change in visibility will be novel—minimization techniques haven't been broadly deployed before—the effects will also be a motivation for new research.

Astronomical eclipses, too, have been a source of inspiration to researchers, perhaps most notably the famous Eddington experiment of 1919 (ironically, for the time of this present writing, in the midst of another global pandemic). Eclipses had long been studied, but the change in visibility of stars, or more precisely, of the observed location of starlight passing the Sun, had not been measured. Arthur Stanley Eddington and Frank Watson Dyson organized expeditions to Principe and Sobral to record the location of the Hyades, a group of stars, during a solar eclipse[81]. The starlight's degree of deflection by the Sun's gravity confirmed Einstein's theory of General Relativity.

Whereas Eddington's team understandably focused on a single group of stars, the DNS community will have millions of resolvers to watch. Eventually, perhaps, minimization will reach a practical maximum. But in the meantime, each resolver will be impacted in its own ways by minimization techniques. Each will also provide unique insights about the global DNS, given the aggregate characteristics of its clients and how they use DNS. Each step along the way is therefore well worth studying. For DNS and Internet protocol researchers, the minimization eclipse is just starting, and the shadows are still partial. DNS resolution is entering the *penumbra*[0].

Acknowledgements

- The idea for this article emerged from multiple conversations with my Verisign colleagues about the history and future of qname minimization. Special thanks to Danny McPherson for his foundational work and strategic direction in this area, and to Duane Wessels and Matt Thomas for their expert technical guidance on both details and data of the techniques.
- Geoff Huston, Chief Scientist of APNIC, Vladimír Čunát and Ladislav Lhotka of CZ.NIC, Puneet Sood of Google, Victoria Risk and her colleagues at ISC, and Benno Overeinder of NLnet Labs all gave generously of their time to review drafts and respond to questions.
- The article would not have reached final form without Zaid AlBanna's leadership in arranging multiple rounds of internal and external reviews, and in coordinating the revisions based on reviewers' helpful feedback. A thank you also to Kim Kelly for her careful technical editing.
- Finally, a thank you to Ole Jacobsen for persevering with IPJ and to IPJ's anonymous reviewers for their important work in reviewing this and many other contributions.

References

[0] Penumbra: A partially shaded area around the edges of a shadow, especially an eclipse. (Source: **`Wiktionary.org`**)

[1] IANA, "Root Servers," **`https://www.iana.org/domains/root/servers`**

[2] Paul V. Mockapetris, "Domain names – implementation and specification," RFC 1035, November 1987.

[3] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, "Protocol Modifications for the DNS Security Extensions," RFC 4035, March 2005.

[4] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, "Resource Records for the DNS Security Extensions," RFC 4034, March 2005.

[5] Ben Laurie, Geoffrey Sisson, Roy Arends, and David Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence," RFC 5155, March 2008.

[6] Burton Kaliski, "Minimum Disclosure: What Information Does a Name Server Need to Do Its Job?" in *Verisign Blog*, March 2015, **`https://blog.verisign.com/security/minimum-disclosure-what-information-does-a-name-server-need-to-do-its-job/`**

[7] Stephane Bortzmeyer, Ralph Dolmans, and Paul Hoffman, "DNS Query Name Minimisation to Improve Privacy," RFC 9156, November 2021.

[8] Stephane Bortzmeyer and Shumon Huque, "NXDOMAIN: There Really Is Nothing Underneath," RFC 8020, November 2016.

[9] Kazunori Fujiwara, Akira Kato, and Warren Kumari, "Aggressive Use of DNSSEC-Validated Cache," RFC 8198, July 2017.

[10] Warren Kumari and Paul Hoffman, "Running a Root Server Local to a Resolver," RFC 8806, June 2020.

[11] Sara Dickinson, Benno Overeinder, Roland van Rijswijk-Deij, and Allison Mankin, "Recommendations for DNS Privacy Service Operators," RFC 8932, October 2020.

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

[13] Geoff Huston, "A Look at DNS Trends and What the Future May Hold," in *CircleID Blog*, October 2020, **https://circleid.com/posts/20201028-a-look-at-dns-trends-and-what-the-future-may-hold/**

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

[15] Mozilla Foundation, "Public Suffix List," **https://publicsuffix.org/**

[16] Geoff Huston, "DNS Query Privacy revisited," in *APNIC Blog*, September 2020, **https://blog.apnic.net/2020/09/11/dns-query-privacy-revisited/**

[17] Basileal Imana, Aleksandra Korolova, and John Heidemann, "Institutional privacy risks in sharing DNS data," *in ANRW '21: Proceedings of the Applied Networking Research Workshop,* pp. 69–75, ACM, July 2021.

[18] Vincent Levigneron, "ENT was here!!!", presented at OARC 25, Dallas, October 2016, **https://indico.dns-oarc.net/event/25/contributions/403/**

[19] Petr Špaček, "Measuring Efficiency of Aggressive Use of DNSSEC-Validated Cache (RFC 8198)," presented at OARC 28, San Juan, March 2018, **https://indico.dns-oarc.net/event/28/contributions/509/**

[20] Sam Weiler and Johan Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing," RFC 4470, April 2006.

[21] R. (Miek) Gieben and W. (Matthijs) Mekking, "Authenticated Denial of Existence in the DNS," RFC 7129, February 2014.

[22] Peter Van Dijk, "DVE-2018-0003: inaccurate NSEC3 answer results in domain unreachability if the resolver applies aggressive negative caching," September 2018, **https://github.com/dns-violations/dns-violations/blob/master/2018/DVE-2018-0003.md**

[23] Peter Van Dijk, "DVE-2021-0001: inaccurate NSEC3 answer results in domain unreachability if the resolver applies aggressive negative caching," June 2021, **https://github.com/dns-violations/dns-violations/blob/master/2021/DVE-2021-0001.md**

[24] Petr Špaček, "Error in DNSSEC implementation on F5 BIG-IP load balancers" October 2019, **https://en.blog.nic.cz/2019/07/10/error-in-dnssec-implementation-on-f5-big-ip-load-balancers/**

[25] Geoff Huston, "NSEC Caching Revisited," presented at OARC 31, Austin, October 2019, **https://indico.dns-oarc.net/event/32/contributions/717/**

[26] Daniel J. Bernstein, "Breaking DNSSEC," presented at *3rd Usenix Workshop on Offensive Technologies* (WOOT'09), August 2009, **https://www.usenix.org/legacy/events/woot09/tech/** Slides: **https://cr.yp.to/talks/2009.08.10/slides.pdf**

[27] Sharon Goldberg, Moni Naor, Dimitrios Papadopoulos, Leonid Reyzin, Sachin Vasant, and Asaf Ziv, "NSEC5: Provably Preventing DNSSEC Zone Enumeration," in *2015 Network and Distributed System Security Symposium*, February 2015.

[28] Warren Kumari and Paul Hoffman, "Decreasing Access Time to Root Servers by Running One on Loopback," RFC 7706, November 2015.

[29] ICANN Office of the Chief Technology Officer, "ICANN's Root Name Service Strategy and Implementation," OCTO-016, October 2020, **https://www.icann.org/en/system/files/files/octo-016-26oct20-en.pdf**

[30] Edward Lewis and Aalfred Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)," RFC 5936, June 2010.

[31] Masataka Ohta, "Incremental Zone Transfer in DNS," RFC 1995, August 1996.

[32] Willem Toorop, Sara Dickinson, Shivan Sahib, Pallavi Aras, and Allison Mankin, "DNS Zone Transfer-over-TLS," RFC 9103, August 2021.

[33] USC/ISI, "LocalRoot – Serve Yourself the Root," **https://localroot.isi.edu/**

[34] Duane Wessels, Piet Barber, Matt Weinberg, Warren Kumari, and Wes Hardaker, "Message Digest for DNS Zones," RFC 8976, February 2021.

[35] Marc Kührer, Thomas Hupperich, Jonas Bushart, and Christian Rossow, "Going wild: Large-scale classification of open DNS resolvers," in *2015 Internet Measurement Conference Proceedings,* pp. 355–368, ACM, October 2015.

[36] Geoff Huston, "DNS Query Privacy revisited," in *APNIC Blog*, September 2020, **https://blog.apnic.net/2020/09/11/dns-query-privacy-revisited/**

[37] "Security/DOH-resolver-policy," in *MozillaWiki*, **https://wiki.mozilla.org/Security/DOH-resolver-policy**

[38] Wouter Bastiaan de Vries, Quirin Scheitle, Moritz Müller, Willem Toorop, Ralph Dolmans, and Roland van Rijswijk-Deij, "A first look at QNAME minimization in the Domain Name System," in *Passive and Active Measurement*, PAM 2019, pp. 147–160, Springer, March 2019.

[39] Vicky Risk, "BIND to Add QNAME Minimization," in *ISC Blog*, March 2018, **https://www.isc.org/blogs/bind-to-add-qname-minimization/**

[40] Knot Resolver, "Knot Resolver 1.0.0 released," May 2016, **https://www.knot-resolver.cz/2016-05-30-knot-resolver-1.0.0.html**

[41] "PowerDNS Recursor 4.3.0 Released," in *PowerDNS Technical Blog*, March 2020, **https://blog.powerdns.com/2020/03/03/powerdns-recursor-4-3-0-released/**

[42] Ralph Dolmans, "Unbound QNAME minimization," presented at OARC 24, Buenos Aires, March 2016, **https://indico.dns-oarc.net/event/22/contributions/332/**

[43] Alexander Harrison, "Cisco Umbrella DNS and QNAME Minimization," **https://support.umbrella.com/hc/en-us/articles/360032551931-Cisco-Umbrella-DNS-and-QNAME-Minimization**

[44] Ólafur Guðmundsson, "Introducing DNS Resolver, 1.1.1.1 (not a joke)," in *The Cloudflare Blog*, April 2018, **https://blog.cloudflare.com/dns-resolver-1-1-1-1/**

[45] "Comcast's Xfinity Internet Service Joins Firefox's Trusted Recursive Resolver Program," in *Firefox News*, June 2020, **https://blog.mozilla.org/en/products/firefox/firefox-news/comcasts-xfinity-internet-service-joins-firefoxs-trusted-recursive-resolver-program/**

[46] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman, "Clouding up the Internet: how centralized is DNS traffic becoming?" in *IMC '20: Proceedings of the 2020 Internet Measurement Conference*, pp. 42–49, ACM, October 2020, **https://dl.acm.org/doi/10.1145/3419394.3423625**

[47] NextDNS, "Privacy Policy," **https://nextdns.io/privacy**

[48] Google Public DNS, "Prepending nonce labels to query names," **https://developers.google.com/speed/public-dns/docs/security?hl=en#nonce_prefixes**

[49] Wouter Bastiaan de Vries, "Improving Anycast with Measurements," PhD dissertation, University of Twente, December 2019, **https://research.utwente.nl/en/publications/improving-anycast-with-measurements**

[50] NLnet Labs,"Qname Minimization," **https://dnsthought.nlnetlabs.nl/#qnamemin**

[51] Geoff Huston, private communications, October 2021.

[52] Matt Thomas, "Maximizing Qname Minimization: A New Chapter in DNS Protocol Evolution," in *Verisign Blog*, September 2020, **https://blog.verisign.com/security/maximizing-qname-minimization-a-new-chapter-in-dns-protocol-evolution/**

[53] Burton Kaliski, "Standardizing Confidentiality Protections for Domain Name System (DNS) Exchanges: Multiple Approaches, New Functionality," in *IEEE Communications Standards Magazine*, September 2021.

[54] Petr Špaček, "NXNSAttack: Upgrade Resolvers to Stop New Kind of Random Subdomain Attack," in *RIPE Labs Blog*, May 2020, **https://labs.ripe.net/author/petr_spacek/nxnsattack-upgrade-resolvers-to-stop-new-kind-of-random-subdomain-attack/**

[55] "Third alpha release of PowerDNS Recursor 4.3.0," in *PowerDNS Technical Blog*, October 2019, **https://blog.powerdns.com/2019/10/29/third-alpha-release-of-powerdns-recursor-4-3-0/**

[56] NLnet Labs, "Unbound RFC Compliance," **https://nlnetlabs.nl/projects/unbound/rfc-compliance/**

[57] ISC, "BIND 9.19.x," **https://gitlab.isc.org/isc-projects/bind9/-/issues/53**

[58] Ray Bellis, "Aggressive NSEC caching in BIND 9.12," in *APNIC Blog*, February 2018, **https://blog.apnic.net/2018/02/06/aggressive-nsec-caching-bind-9-12/**

[59] CZ.NIC Labs, "Knot Resolver 2.0.0 (2018-01-31)," January 2018, **https://knot-resolver.readthedocs.io/en/stable/**

[60] "First Beta Release of PowerDNS Recursor 4.5.0," in *PowerDNS Technical Blog*, March 2021, **https://blog.powerdns.com/2021/03/26/first-beta-release-of-powerdns-recursor-4-5-0/**

[61] NLnet Labs, "Aggressive NSEC," **https://unbound.docs.nlnetlabs.nl/en/latest/topics/privacy/aggressive-nsec.html**

[62] Ralph Dolmans, "Aggressive use of the DNSSEC-Validated cache in Unbound," in *NLnet Labs Blog*, April 2018, **https://medium.com/nlnetlabs/aggressive-use-of-the-dnssec-validated-cache-in-unbound-1ab3e315d13f**

[63] Google Public DNS, **https://developers.google.com/speed/public-dns/docs/security#dnssec**

[64] Edward Winstead, "Running a local copy of the root zone," presented at APRICOT 2017/APNIC 43, February 2017, **https://2017.apricot.net/assets/files/APIC674/localdnszone_1488009521.pdf**

[65] CZ.NIC Labs, "Root on loopback (RFC 7706)," **https://knot-resolver.readthedocs.io/en/v5.0.1/modules-rfc7706.html**

[66] ICANN Security and Stability Advisory Committee, "Invalid Top Level Domain Queries at the Root Level of the Domain Name System," SAC045, November 2010, **https://www.icann.org/en/system/files/files/sac-045-en.pdf**

[67] DNS-OARC, "Day In The Life of the Internet," **https://www.dns-oarc.net/oarc/data/ditl**

[68] Verisign, "Workshop on Root Causes and Mitigation of Name Collisions (WPNC)–March 2014."

[69] US-CERT, "WPAD Name Collision Vulnerability," Alert TA-144A, revised October 2016, **https://us-cert.cisa.gov/ncas/alerts/TA16-144A**

[70] Qi Alfred Chen, Eric Osterweil, Matthew Thomas, and Z. Morley Mao, "MitM attack by name collision: Cause analysis and vulnerability assessment in the new gTLD era," in *2016 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2016, **https://ieeexplore.ieee.org/abstract/document/7546529**

[71] Qi Alfred Chen, Matthew Thomas, Eric Osterweil, Yulong Cao, and Jie You, "Client-side Name Collision Vulnerability in the New gTLD Era: A Systematic Study," in CCS '17: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 941–956, ACM, October 2017, **https://doi.org/10.1145/3133956.3134084**

[72] Carnegie-Mellon University Software Engineering Institute, "Microsoft Windows domain-configured client Group Policy fails to authenticate servers," Vulnerability Note VU#787252, February 2015, **https://www.kb.cert.org/vuls/id/787252**

[73] Matt Thomas, "Verisign Outreach Program Remediates Billions of Name Collision Queries," in *Verisign Blog*, January 2021, **https://blog.verisign.com/domain-names/verisign-outreach-program-remediates-billions-of-name-collision-queries/**

[74] Roy Arends and Nicolas Antoniello, ICANN Office of the CTO, "Hyperlocal Root Zone Technical Analysis," OCTO-027, August 2021, **https://www.icann.org/en/system/files/files/octo-027-25aug21-en.pdf**

[75] Matt Thomas, "Chromium's impact on root DNS traffic," in *APNIC Blog*, August 2020, **https://blog.apnic.net/2020/08/21/chromiums-impact-on-root-dns-traffic/**

[76] Duane Wessels and Christian Huitema, "More Mysterious Root Query Traffic from a Large Recursive Operator," presented at OARConline 35a, September 2021, **https://indico.dns-oarc.net/event/39/contributions/864/**

[77] Geoff Huston, "Another DNS OARC meeting," in *APNIC Blog*, September 2021, **https://blog.apnic.net/2021/09/14/another-dns-oarc-meeting/**

[78] Florian Weimer. "Passive DNS replication," in *FIRST Conference on Computer Security Incident Handling*, Volume 98, 2005, **https://www.first.org/resources/papers/conference2005/florian-weimer-paper-1.pdf**

[79] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)," Internet Draft, work in progress, October 2022, **draft-ietf-dnsop-svcb-https**.

[80] Tommy Pauly, "Encrypted DNS support in iOS and macOS," IETF ADD Working Group mailing list, June 2020, **https://mailarchive.ietf.org/arch/msg/add/MbOOWPVHRHM_wvbKhfHuzUTwimI**

[81] Watson Dyson, Arthur Stanley Eddington, and Charles Rundle Davidson, "A Determination of the Deflection of Light by the Sun's Gravitational Field, from Observations Made at the Total Eclipse of May 29, 1919," *Philosophical Transactions of the Royal Society of London, Series A, Containing Papers of a Mathematical or Physical Character*, 220(571–581), pp. 291–333, 1920. **https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1920.0009**

Also available from:
https://w.astro.berkeley.edu/~kalas/labs/documents/dyson1920.pdf

BURTON S. KALISKI JR. (**bkaliski@verisign.com**) is senior vice president and chief technology officer of Verisign. He leads Verisign's long-term research program and is responsible for the company's industry standards engagements, university collaborations, and technical community programs. He previously served as the founding director of the EMC Innovation Network, as vice president of research at RSA Security, and as the founding scientist of RSA Laboratories, where his contributions included the development of the *Public-Key Cryptography Standards* (PKCS). He received a doctorate, master's degree, and bachelor's degree in computer science from the Massachusetts Institute of Technology.

KINDNS

by Adiel Akplogan, ICANN

In September 2022, the *Internet Corporation for Assigned Names and Numbers* (ICANN) launched **`https://kindns.org`** to support its *Knowledge-Sharing and Instantiating Norms for Domain Name System and Naming Security* (KINDNS) initiative. KINDNS was developed to improve DNS operations by promoting voluntary adherence to a clear set of security best practices tailored to authoritative and recursive DNS operators.

This initiative aligns with ICANN's strategic goal to "Strengthen DNS coordination in partnership with the DNS stakeholders to improve the shared responsibility for upholding the security and stability of the DNS." In other words, ICANN plans to actively promote DNS ecosystem security and relevant best practices. The KINDNS initiative is one of many programs and projects ICANN supports to help make the DNS safer.

The Domain Name System plays a crucial role in connecting users to services on the Internet. Like the Internet itself, the underlying protocols or rules that govern DNS operation are open. This is one of the greatest strengths of the DNS and the Internet: These open protocols are responsible for connecting billions of devices instantaneously all over the world.

This strength can also be a weakness. The DNS was not designed for security. Actors may snoop on DNS traffic, forge DNS traffic, and engage in denial-of-service attacks on DNS operations, among other activities. Similarly, the security systems and best practices of the DNS, which support the Internet's operation, are characteristically open and voluntary. A clear example of this is the uneven global deployment of the *Domain Name System Security Extensions* (DNSSEC), a security enhancement specification developed by the *Internet Engineering Task Force* (IETF) more than 20 years ago.

DNS security challenges, however, are not unique. Key to Internet security, or security issues in general, is the necessity to coordinate behaviors across systems. Security challenges call for collective action and the voluntary adherence to a set of ever-evolving behaviors and technologies.

ICANN is uniquely positioned, in close collaboration with its many partners and peers, to help in collectively mitigating specific forms of DNS security threats, that fall within its mission (see below for more details). The organization's greatest strength, in many ways, is its partnerships, active community participation, and global engagement activities. This network of technical organizations and experts from various backgrounds can provide a formidable tool in helping make the global DNS safer.

What is KINDNS?

KINDNS is an ICANN initiative tailored to authoritative and recursive DNS operators to promote voluntary adherence to a clear set of security best practices. A challenge facing DNS security is implementing and maintaining security at the same level for all authoritative and recursive operators in the DNS ecosystem. Smaller operators struggle to keep abreast of the latest security measure improvements, while large operators may implement only measures which help them achieve their professional business goals. As a result, a patchwork of varying security practices among DNS operators has led to weaknesses that malicious actors may find and use.

Currently, KINDNS has three areas of focus:

- Promoting the adoption of DNS security practices through the operator community. This includes maintaining a dynamic information portal that promotes KINDNS practices, helps operators to self-assess their practices and offers guidelines on how to implement them.
- Soliciting and gathering feedback on the KINDNS guidelines to refine and identify areas of improvement and emerging best practices that may be candidates for future additions to the framework.
- Developing advanced tools for operators to conduct self-assessments and an observatory platform around key DNS security indicators that can help measure and assess the impact of KINDNS.

KINDNS Portal

ICANN has worked with its community to develop a baseline level of security operations, a relatively small set of mutually agreed practices that operators of any size can easily implement. ICANN published the initial version of KINDNS in September 2022 with its own dedicated website: `https://kindns.org`

Voluntary Self-Assessment Form

To learn more about the initiative, KINDNS offers operators a simple self-assessment tool to check their current security practices and offers suggestions where they could improve. The current version of the self-assessment tool does not run any code on an operator's server or make any real-time measurements. KINDNS only asks questions and generates a compliance report upon completion of the survey. The form is found here: `https://kindns.org/assessments-tools`

KINDNS Guidelines and Practices

The guidelines and practices promoted by KINDNS were designed to help operators identify and implement critical security best practices. ICANN, however, does not view the KINDNS initiative as the only source of information on DNS security. On the contrary, it encourages operators to also check and work in accordance with the security dictated by the construct of their own infrastructure.

Currently, the KINDNS Framework covers the following categories:

Guidelines for Authoritative Server Operations

TLD and Critical Zones and Other Second-Level Domains (SLD) *Zones*: There are two types of best practices for operators of authoritative servers: DNS security and DNS availability and resilience. To learn more about these guidelines see:

https://kindns.org/critical-zones

and

https://kindns.org/other-sld-zones

Guidelines for Recursive Server Operators

These guidelines are tailored for three specific categories of recursive server operators. Depending on a company's policy or business practice, it may be operating one or more of these types of resolvers. These three categories include:

- *Private Resolvers:* these are not publicly accessible and cannot be reached over the open Internet. They are typically found in corporate networks or other restricted-access networks. Private resolvers in some cases are part of a trusted computing domain (for example, *Active Directory*).
- *Shared Private Resolver Operators:* these are typically *Internet Service Providers* (ISPs) or similar hosting service providers. They offer DNS resolution services to their customers (including for mobile, cable, DSL, fiber residential and commercial users, and hosted servers and applications).
- *Public resolvers:* this category includes both open and closed public resolvers. Closed public resolvers are typically commercial DNS filtering or scrubbing services. These service providers are typically not ISPs and the clients sending queries to them are located on remote networks. Note, some operators of closed public resolvers may also offer a free tier service, which also makes them open public resolvers.

To learn more about these guidelines visit:

https://kindns.org/recursive-server-operators

Guidelines for Platform Hardening

KINDNS recommends that all operators pay careful attention to practices for hardening the platforms their DNS services use. There are three types of hardening practices:

- *Network Security:* these best practices are aimed at preventing unauthorized network access to DNS servers and ensuring internal traffic does not leak onto other networks.
- *Host and Service Security:* these best practices are aimed at improving the security of hosts running DNS services to reduce the likelihood of a host compromise, denial-of-service attack or other attack.
- *Customer-Facing Portal and Service Security:* these practices are aimed at supporting the security needs of customers.

To learn more about these guidelines, visit:

https://kindns.org/platform-hardening

Conclusion

The website launch marks the completion of this initial phase of the KINDNS initiative, where ICANN's Technical Engagement team worked closely with the operator community and DNS experts to identify and document DNS security threats and their mitigation measures, which are the basis for the current guidelines. We hope and expect to evolve them as the DNS and the Internet continue to evolve. We invite anyone interested in participating in this initiative to join the KINDNS mailing list at:
https://mm.icann.org/mailman/listinfo/kindns-discuss

If you have any questions, please contact the KINDNS team at:
kindns-info@icann.org

ADIEL AKPLOGAN is Vice President, Technical Engagement at ICANN. With more than 25 years' experience in the ICT industry (20 specifically in the Internet Technology Industry), Adiel previously served as CEO for AFRINIC (The African Network Information Centre), IT Director for Symbol Technology in France (2001–2003) and Director of New Technology at CAFÉ Informatique in Togo (1994–2000). He earned a graduate degree in Electrical Engineering and holds a Master's degree in E-Business and New Technology Management from Paris Graduate School of Management. Recognized as one of the Internet technology pioneers in Africa, he has contributed to technical capacity building and deployment of some of the first private Internet Service Providers in Africa from 1996 to 1999. He can be reached at:
adiel.akplogan@icann.org

Supporters and Sponsors

Supporters

Internet Society

CISCO

Diamond Sponsors

Your logo here!

Ruby Sponsors

ICANN

Sapphire Sponsors

APNIC FOUNDATION

Emerald Sponsors

Akamai

amsix

APRICOT Asia Pacific Regional Internet Conference on Operational Technologies

DE-CIX

EQUINIX

Google

identity digital

JPRS

JUNIPER NETWORKS

lacnic

LinkedIn

linx

netskope

NSRC Network Startup Resource Center

NTT Communications

RIPE NCC RIPE NETWORK COORDINATION CENTRE

TEAM CYMRU

VERISIGN

WIDE PROJECT

Corporate Subscriptions

AFRINIC The Internet Numbers Registry for Africa

COMCAST

DNS-OARC

Edgio

IPXO

ISC Internet Systems Consortium

IWL

qa cafe

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

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**

Kjetil Aas
Fabrizio Accatino
Michael Achola
Martin Adkins
Melchior Aelmans
Christopher Affleck
Scott Aitken
Jacobus Akkerhuis
Antonio Cuñat Alario
William Allaire
Nicola Altan
Shane Amante
Marcelo do Amaral
Matteo D'Ambrosio
Selva Anandavel
Jens Andersson
Danish Ansari
Finn Arildsen
Tim Armstrong
Richard Artes
Michael Aschwanden
David Atkins
Jac Backus
Jaime Badua
Bent Bagger
Eric Baker
Fred Baker
Santosh Balagopalan
William Baltas
David Bandinelli
Benjamin Barkin-Wilkins
Feras Batainah
Michael Bazarewsky
David Belson
Richard Bennett
Matthew Best
Hidde Beumer
Pier Paolo Biagi
Arturo Bianchi
John Bigrow
Orvar Ari Bjarnason
Tyson Blanchard
Axel Boeger
Keith Bogart
Mirko Bonadei
Roberto Bonalumi
Lolke Boonstra
Julie Bottorff Photography
Gerry Boudreaux
Leen de Braal
Kevin Breit
Thomas Bridge
Ilia Bromberg
Lukasz Bromirski
Václav Brožík
Christophe Brun
Gareth Bryan
Ron Buchalski
Paul Buchanan
Stefan Buckmann
Caner Budakoglu
Darrell Budic
BugWorks
Scott Burleigh
Chad Burnham
Colin Butcher
Jon Harald Bøvre
Olivier Cahagne
Antoine Camerlo
Tracy Camp
Ignacio Soto Campos
Brian Candler
Fabio Caneparo
Roberto Canonico
David Cardwell
Richard Carrara
John Cavanaugh
Lj Cemeras
Dave Chapman
Stefanos Charchalakis
Molly Cheam
Greg Chisholm
David Chosrova
Marcin Cieslak
Lauris Cikovskis
Brad Clark
Narelle Clark
Horst Clausen
James Cliver
Guido Coenders
Joseph Connolly
Steve Corbató
Brian Courtney
Beth and Steve Crocker
Dave Crocker
Kevin Croes
John Curran
André Danthine
Morgan Davis
Jeff Day
Julien Dhallenne
Freek Dijkstra
Geert Van Dijk
David Dillow
Richard Dodsworth
Ernesto Doelling
Michael Dolan
Eugene Doroniuk
Karlheinz Dölger
Michael Dragone
Joshua Dreier
Lutz Drink
Aaron Dudek
Dmitriy Dudko
Andrew Dul
Joan Marc Riera Duocastella
Pedro Duque
Holger Durer
Mark Eanes
Andrew Edwards
Peter Robert Egli
George Ehlers
Peter Eisses
Torbjörn Eklöv
Y Ertur
ERNW GmbH
ESdatCo
Steve Esquivel
Jay Etchings
Mikhail Evstiounin
Bill Fenner
Paul Ferguson
Ricardo Ferreira
Kent Fichtner
Armin Fisslthaler
Michael Fiumano
The Flirble Organisation
Gary Ford
Jean-Pierre Forcioli
Susan Forney
Christopher Forsyth
Andrew Fox
Craig Fox
Fausto Franceschini
Valerie Fronczak
Tomislav Futivic
Laurence Gagliani
Edward Gallagher
Andrew Gallo
Chris Gamboni
Xosé Bravo Garcia
Osvaldo Gazzaniga
Kevin Gee
Greg Giessow
John Gilbert
Serge Van Ginderachter
Greg Goddard
Tiago Goncalves
Ron Goodheart
Octavio Alfageme Gorostiaga
Barry Greene
Jeffrey Greene
Richard Gregor
Martijn Groenleer
Geert Jan de Groot
Ólafur Guðmundsson
Christopher Guemez
Gulf Coast Shots
Sheryll de Guzman
Rex Hale
Jason Hall
James Hamilton
Darow Han
Handy Networks LLC
Stephen Hanna
Martin Hannigan
John Hardin
David Harper
Edward Hauser
David Hauweele
Marilyn Hay
Headcrafts SRLS
Hidde van der Heide
Johan Helsingius
Robert Hinden
Asbjørn Højmark
Damien Holloway
Alain Van Hoof
Edward Hotard
Bill Huber
Hagen Hultzsch
Kauto Huopio
Kevin Iddles
Mika Ilvesmaki
Karsten Iwen
Joseph Jackson
David Jaffe
Ashford Jaggernauth
Thomas Jalkanen
Martijn Jansen
Jozef Janitor
John Jarvis
Dennis Jennings
Edward Jennings
Aart Jochem
Nils Johansson
Brian Johnson
Curtis Johnson
Richard Johnson
Jim Johnston
Jonatan Jonasson
Daniel Jones
Gary Jones
Jerry Jones
Michael Jones
Amar Joshi
Javier Juan
David Jump
Anders Marius Jørgensen
Merike Kaeo
Andrew Kaiser
Christos Karayiannis
Daniel Karrenberg
David Kekar
Stuart Kendrick
Robert Kent
Jithin Kesavan
Jubal Kessler
Shan Ali Khan
Nabeel Khatri
Dae Young Kim
William W. H. Kimandu
John King
Russell Kirk
Gary Klesk
Anthony Klopp
Henry Kluge
Michael Kluk
Andrew Koch
Ia Kochiashvili
Carsten Koempe
Richard Koene
Alexader Kogan
Matthijs Koot
Antonin Kral
Robert Krejčí
Mathias Körber
John Kristoff
Terje Krogdahl
Bobby Krupczak
Murray Kucherawy
Warren Kumari
George Kuo
Dirk Kurfuerst
Darrell Lack
Andrew Lamb
Richard Lamb
Yan Landriault
Edwin Lang
Sig Lange
Markus Langenmair
Fred Langham
Tracy LaQuey Parker
Alex Latzko
Jose Antonio Lazaro Lazaro
Antonio Leding
Rick van Leeuwen

Simon Leinen
Robert Lewis
Christian Liberale
Martin Lillepuu
Roger Lindholm
Link Light Networks
Chris and Janet Lonvick
Sergio Loreti
Eric Louie
Adam Loveless
Josh Lowe
Guillermo a Loyola
Hannes Lubich
Dan Lynch
David MacDuffie
Sanya Madan
Miroslav Madić
Alexis Madriz
Carl Malamud
Jonathan Maldonado
Michael Malik
Tarmo Mamers
Yogesh Mangar
John Mann
Bill Manning
Harold March
Vincent Marchand
Normando Marcolongo
Gabriel Marroquin
David Martin
Jim Martin
Ruben Tripiana Martin
Timothy Martin
Carles Mateu
Juan Jose Marin Martinez
Ioan Maxim
David Mazel
Miles McCredie
Brian McCullough
Joe McEachern
Alexander McKenzie
Jay McMaster
Mark Mc Nicholas
Olaf Mehlberg
Carsten Melberg
Kevin Menezes
Bart Jan Menkveld
Sean Mentzer
William Mills
David Millsom
Desiree Miloshevic
Joost van der Minnen
Thomas Mino
Rob Minshall
Wijnand Modderman-Lenstra
Mohammad Moghaddas
Roberto Montoya
Charles Monson
Andrea Montefusco
Fernando Montenegro
Joel Moore
John More
Maurizio Moroni
Brian Mort
Soenke Mumm
Tariq Mustafa
Stuart Nadin
Michel Nakhla
Mazdak Rajabi Nasab
Krishna Natarajan
Naveen Nathan
Darryl Newman
Thomas Nikolajsen
Paul Nikolich
Travis Northrup
Marijana Novakovic
David Oates
Ovidiu Obersterescu
Tim O'Brien
Mike O'Connor
Mike O'Dell
John O'Neill
Jim Oplotnik
Carl Önne
Packet Consulting Limited
Carlos Astor Araujo Palmeira
Alexis Panagopoulos
Gaurav Panwar
Chris Parker
Manuel Uruena Pascual
Ricardo Patara
Dipesh Patel
Alex Parkinson
Craig Partridge
Dan Paynter
Leif Eric Pedersen
Rui Sao Pedro
Juan Pena
Chris Perkins
Michael Petry
Alexander Peuchert
David Phelan
Harald Pilz
Derrell Piper
Rob Pirnie
Marc Vives Piza
Jorge Ivan Pincay Ponce
Victoria Poncini
Blahoslav Popela
Andrew Potter
Ian Potts
Eduard Llull Pou
Tim Pozar
David Raistrick
Priyan R Rajeevan
Balaji Rajendran
Paul Rathbone
William Rawlings
Mujtiba Raza Rizvi
Bill Reid
Petr Rejhon
Robert Remenyi
Rodrigo Ribeiro
Glenn Ricart
Justin Richards
Rafael Riera
Mark Risinger
Fernando Robayo
Michael Roberts
Gregory Robinson
Ron Rockrohr
Carlos Rodrigues
Magnus Romedahl
Lex Van Roon
Marshall Rose
Alessandra Rosi
David Ross
William Ross
Boudhayan Roychowdhury
Carlos Rubio
Rainer Rudigier
Timo Ruiter
RustedMusic
Babak Saberi
George Sadowsky
Scott Sandefur
Sachin Sapkal
Arturas Satkovskis
PS Saunders
Richard Savoy
John Sayer
Phil Scarr
Gianpaolo Scassellati
Elizabeth Scheid
Jeroen Van Ingen Schenau
Carsten Scherb
Ernest Schirmer
Benson Schliesser
Philip Schneck
James Schneider
Peter Schoo
Dan Schrenk
Richard Schultz
Timothy Schwab
Roger Schwartz
SeenThere
Scott Seifel
Paul Selkirk
Yury Shefer
Yaron Sheffer
Doron Shikmoni
Tj Shumway
Jeffrey Sicuranza
Thorsten Sideboard
Greipur Sigurdsson
Fillipe Cajaiba da Silva
Andrew Simmons
Pradeep Singh
Henry Sinnreich
Geoff Sisson
John Sisson
Helge Skrivervik
Terry Slattery
Darren Sleeth
Richard Smit
Bob Smith
Courtney Smith
Eric Smith
Mark Smith
Tim Sneddon
Craig Snell
Job Snijders
Ronald Solano
Asit Som
Ignacio Soto Campos
Evandro Sousa
Peter Spekreijse
Thayumanavan Sridhar
Paul Stancik
Ralf Stempfer
Matthew Stenberg
Martin Štěpánek
Adrian Stevens
Clinton Stevens
John Streck
Martin Streule
David Strom
Colin Strutt
Viktor Sudakov
Edward-W. Suor
Vincent Surillo
Terence Charles Sweetser
T2Group
Roman Tarasov
David Theese
Douglas Thompson
Kerry Thompson
Lorin J Thompson
Fabrizio Tivano
Peter Tomsu Fine Art Photography
Joseph Toste
Rey Tucker
Sandro Tumini
Angelo Turetta
Michael Turzanski
Phil Tweedie
Steve Ulrich
Unitek Engineering AG
John Urbanek
Martin Urwaleck
Betsy Vanderpool
Surendran Vangadasalam
Ramnath Vasudha
Philip Venables
Buddy Venne
Alejandro Vennera
Luca Ventura
Scott Vermillion
Tom Vest
Peter Villemoes
Vista Global Coaching & Consulting
Dario Vitali
Rüdiger Volk
Jeffrey Wagner
Don Wahl
Michael L Wahrman
Laurence Walker
Randy Watts
Andrew Webster
Tim Weil
Jd Wegner
Westmoreland Engineering Inc.
Rick Wesson
Peter Whimp
Russ White
Jurrien Wijlhuizen
Derick Winkworth
Pindar Wong
Makarand Yerawadekar
Phillip Yialeloglou
Janko Zavernik
Bernd Zeimetz
Muhammad Ziad Ziayuddin
Tom Zingale
Jose Zumalave
Romeo Zwart
廖明沂.

Follow us on Twitter and Facebook

@protocoljournal

https://www.facebook.com/newipj

The Internet Protocol Journal
Link Fulfillment
7650 Marathon Dr., Suite E
Livermore, CA 94550

CHANGE 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