IIJ Public DNSサービスについて

2.1 はじめに

IIJ Public DNSがベータ版サービスとして5月にリリースされました。DNS over TLS（以下 DoT）、DNS over HTTPS（DoH）「だけ」、すなわち通常のUDP/TCPによる名前解決をサポートせず、そしてIIJのユーザだけでなく誰でも使っていただけるキャッシュDNSサービスです。

本稿では、DoT/DoHが通常のDNSとどのように異なるか紹介すると共に、サービスを提供するに当たって検討したポイントと今後の課題について解説します。

2.2 DoT/DoH とは

2.2.1 DNSとプライバシー

DNSに登録される情報は広く公開されるのが前提です。そのため、DNSのセキュリティといえば、長い間「改ざんされないこと」すなわち完全性の確保が主眼であり、「盗聴されないこと」すなわち機密性の確保に重きが置かれることはありませんでした。

しかし、2013年のいわゆるスノーデン事件で、アメリカ国家安全保障局（NSA）による大規模な通信監視・情報収集活動PRISMの存在が告発されました。IETFはこれを受けて、"Pervasive Monitoring Is an Attack"（広範な監視は攻撃である）と宣言し（RFC7258）、今後策定されるプロトコルは広域監視に耐え得る設計であることが求められるようになりました。

PRISMの監視対象にDNSが含まれていたことも明らかになり、IETFでは新設のDPRIVE（DNS PRIVate Exchange）ワーキンググループにより、これまでおざなりにされてきたDNSのプライバシー確保の仕組みが検討されることになりました。DPRIVE WGではQname Minimisation（RFC7816）、EDNS（0）padding option（RFC7830、RFC8467）などDNSに対する様々なプロトコル拡張・修正を行いましたが、中でも比重の大きかったのがトランスポート暗号化です。

2.2.2 DNSトランスポート暗号化

従来のDNSは下位プロトコル（トランスポート）として主にUDPを、補助的にTCPを使います。しかし、素のUDP/TCPやその上に乗るDNSに機密性確保の機能はなく、通信が平文のまま行われるため盗聴も容易です。そこで、DNSと下位層の間に暗号化の層を置いて保護することにしたのです。

暗号化の層として様々なものが提案され、現在までにTLSを利用するDNS over TLS（RFC7858）、DTLSを利用するDNS over DTLS（RFC8094）、HTTPSを利用するDNS over HTTPS（RFC8484）が標準化されています。他にQUICを利用するDNS over QUICのドラフトがIETFに提出され議論が始まっています。また、同じく議論中のHTTP/3が標準化されると、DoHも自動的にHTTP/3に対応することになります（図-1）。

これら様々な暗号化層はどれかに一本化されるのではなく、現状ではユーザの都合に合わせて好きなものを選んで使えば良いことになっていますが、乱立してしまうとそれはそれで不便なので、将来的にはいくつかを残して残りは非推奨になることも十分考えられます（現状でもDNS over DTLSは仕様だけがあって実装は存在しておらず、今後使えるようになる可能性は小さいでしょう）。

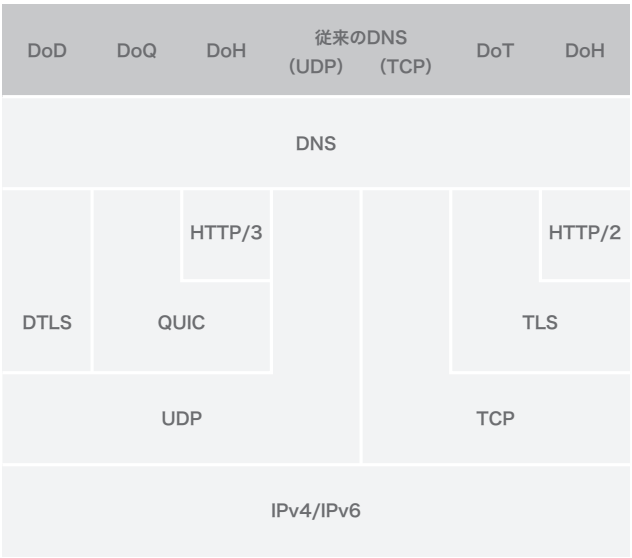

図-1 DNSのトランスポートプロトコル

2.2.3 トランスポート暗号化とDNSSEC

ところで、DNSには電子署名でDNS情報の真正性を検証するDNSSECという仕組みが既にあります。なぜDNSSECがあるのに新たにトランスポート暗号化が必要になるのでしょうか。また、トランスポート暗号化により、今後DNSSECは不要になるのでしょうか。

この問いに答える前に、まずトランスポート暗号化の使われる範囲について解説します。DNSは、ユーザがDNSの大本の情報を登録したサーバ(権威サーバ)に直接問い合わせるのではなく、ISPなどが提供するキャッシュサーバに情報を問い合わせ、キャッシュサーバが権威サーバへの問い合わせを行う構成が一般的です。

トランスポート暗号化が行われるのは、現状ではユーザとキャッシュサーバの間だけです。キャッシュサーバと権威サーバの間は暗号化されず、従来のDNSがそのまま使われます。

一般に暗号化は完全性と機密性のどちらも保証しますが、DNSのトランスポート暗号化についていえば、その範囲はユーザとキャッシュサーバの間に限られます。キャッシュサーバと権威サーバの間は暗号化されない従来のDNSが使われるため、キャッシュサーバが入手した情報の完全性の保証はなく、それを暗号で保護しても完全性は得られません。つまり、一般的な暗号化とは異なり、DNSトランスポートの暗号化はユーザとキャッシュサーバの間の機密性だけが保証されるプロトコルということになります(図-2)。

一方、DNSSECは「改ざんされないこと」を目的に導入されたものです。電子署名を使うことで完全性を保証しますが、通信自体は平文で、機密性はありません。

つまりトランスポート暗号化とDNSSECはどちらもDNSのセキュリティ向上のための仕組みですが、一方は機密性の保証に特化して完全性がなく、もう一方は完全性の保証に特化して機密性がないという相補的な関係になっていて、一方でもう一方を置き換えることはできません。それぞれが守るものが異なるので、トランスポート暗号化があるからDNSSECは要らないということにはならず、その逆もありません。

2.3 IIJ Public DNSサービスとDoT/DoH

DoT/DoHは下位層のトランスポートプロトコルが異なるだけで、DNSのプロトコル自体は従来のものと変わりません。しかし、サービスとして提供するに当たっては乗り越えなければならない壁がいくつかありました。ひとつひとつ見ていきましょう。

2.3.1 TCPの壁

DoT/DoHが従来のDNSと大きく異なるのは、TLSで暗号化されるということ以前に、すべてのやりとりがTCPになることです。従来のDNSはトランスポートとしてTCPとUDPのどちらも使えますが、主に使われるのはUDPで、TCPが使われるケースは限定的です。

TCPは上位層のプロトコルのやりとりが始まる前にセッション確立のための処理を行い、確立後も送信されたパケットが受信

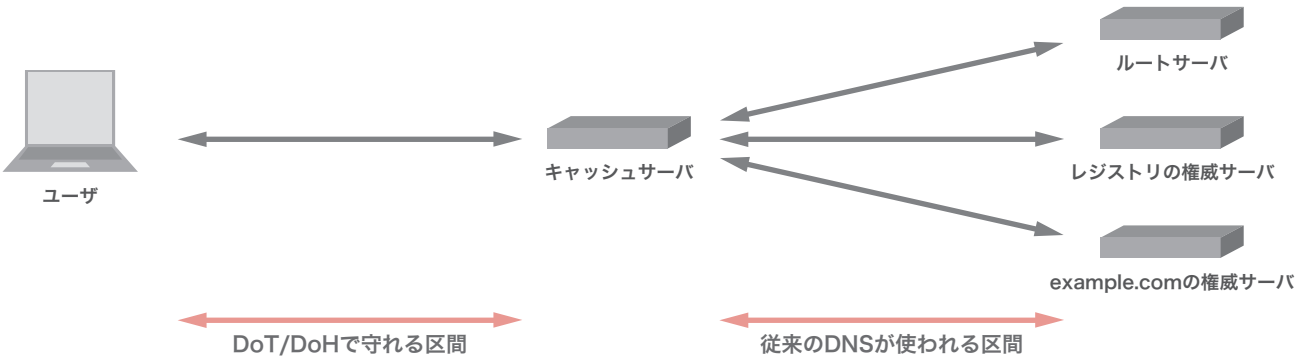

図-2 トランスポート暗号化のスコープ

されたかどうかチェックして必要に応じて再送するなど、通信の信頼性を確保するために様々な処理を行っています。

DNSはやりとりされるデータが非常に小さく、問い合わせ、応答ともにほとんどの場合で数百バイト以上になることはありません。このような用途でTCPを使うと、実際のDNSメッセージのやりとりよりも、セッションの確立・終了処理にかかるやりとりが大半を占めることになり、非常に効率が悪くなってしまいます。負荷の問題ならばサーバの数を増やすという富豪的なアプローチでも解決できますが、パケットの往復回数が増えることによるレイテンシ(遅延)の増大はどうしようもありません。

UDPはそのような仕組みを持たない分、単純・高速で、DNSやNTPなど小さなパケットのやりとりで完結するプロトコルで多く使われてきました。その一方で信頼性は低く、キャッシュポイズニングやDNS amp攻撃など、これまでに見つかったDNSに対する攻撃手法の多くは、DNSというプロトコル自体の問題というよりは、下位層にUDPを利用していることに起因しています。

TCPを使うことでこういった攻撃手法は成立しなくなる、ないしは大幅に脅威が減じられることは分かっているのですが、小さなパケットを大量にやりとりするというDNSの性質上、オーバーヘッドの非常に大きなTCPを主に利用する方向に転換することはこれまでできずにいました。2008年、これまで知られていた手法に比べてはるかに効率よくキャッシュポイズニングが可能なKaminsky Attack[*1]が公表された時でさえ、TCPに乗り換えることなく、ソースポートランダマイゼーション[*2]という対症療法でUDPのままなんとか乗り切ったぐらいです。

それほどTCPのオーバーヘッドを嫌っていたDNSですが、TLSが要求されることになると、必然的にTCPも必要になります。UDPを利用するDNS over DTLSもありますが、オーバーヘッドが大きいという点ではTCPとさほど変わらず、そもそも仕様だけの存在で実装が存在しないので誰も使えません。

とはいえ、安全なキャッシュDNSサービスを提供するための基盤とするなら、これまでの常識を捨てて考えなければなりません。IIJ Public DNSサービスがアクセス制限せず世界中の誰にも使っていただける"public"なDNSサービスとして提供できるのも、信頼性の低いUDP上のDNSを使わないことにして、DNS ampなどの攻撃の踏み台にされる懸念を払拭できたことが大きな要因となっており、TCPであることの利点にも積極的に目を向けなければなりません。

2.3.2 TLSの壁

DoT/DoHでは、UDPが一切使われずTCPになってオーバーヘッドが大きくなった上、更に暗号化のTLSのレイヤーがその上に載ります。TLSはHTTPSなど広く使われている技術ではありますが、決して処理が軽いわけではなく、DNSのように小さなデータを高速でやりとりする必要のあるプロトコルでは大幅な性能劣化をもたらします。

UDPによる従来のDNSと比べて大幅にパフォーマンスが劣化するということは理屈では分かるものの、実際どの程度劣化するのか調べるには測定のための道具が必要です。TCPによるDNSは、使われる場面が限定的だっただけで以前から存在していたので道具もあります。しかしDNS over TLSはまったく新しいものです。パフォーマンスを測定するための道具も満足なものがないのです。どれだけの劣化があるのか、どれだけの負荷があるのかを計測し、サービスとして提供するに当たりどれだけの設備を用意する必要があるのか見積もるために、我々はパフォーマンス測定ツールから開発する必要がありました。

TLSは非常に重い処理であるが故に、以前接続した際の情報を再利用してセッションを再開したり(TLS session resumption)、最新のTLS 1.3ではハンドシェイク手順の最初にやりとりされるClientHello/ServerHelloの中にアプリケーションのデータを入れる(0-RTT)などの方法でオーバーヘッドを小さくしたりといったオプションが利用できるようになっています。

---

*1 例えば、JPRS トピックス＆コラム、「新たなＤＮＳキャッシュポイズニングの脅威～カミンスキー・アタックの出現～」(https://jprs.jp/related-info/guide/009.pdf)など。

*2 クライアントが問い合わせの際に用いるソースポートをランダムにすることで、攻撃者がパケットを偽造するに当たって推測を的中させなければならない要素を増やし、攻撃の成功確率を下げる手法。

ただ、TLSのプロトコルで利用できる機能であっても、アプリケーションがそれを活用しなければ意味がありません。これらのオプションをフルに使いこなさないと大規模環境で実用的に動かすのは困難です。

IIJ Public DNSで利用しているのはオランダのNLnet Labsが開発しているUnboundというDNS実装で、これはかなり古く、DoTがIETF draftになる以前からTLSに対応していました。しかし、UnboundのTLS対応を調査した結果、こういったTLSのオーバーヘッドを減らすための仕組み、具体的にはTLSセッション再開機能が欠けていることが分かりました。パフォーマンス測定の結果でも十分な性能を出せていません。また、利用する暗号アルゴリズムによってもパフォーマンスが大きく変化しますが、これはソースコードに埋め込まれていて設定では変えられません。そのため、IIJではこれらに対応する機能を実装しています。成果はNLnet Labsにフィードバックされ、既に最新版では我々のコードが取り込まれた状態で配布されています。

TLSの壁はパフォーマンスの問題以外にもう1つあります。暗号化されていることそれ自体です。

DNSの安定運用のためには、異常な問い合わせが多数来ていないか、問い合わせは異常でないのに応答で異常なものが増えていないかなどの統計情報を取得し、もし異常があればそれを調査・対応できる仕組みが欠かせません。従来のDNSでは、こういった統計情報取得やトラブルシューティングはDNSサーバ自身で行うのではなく、DNSパケットをキャプチャすることにより実施するケースがほとんどでした。DNSサーバとは独立に処理を行えるため、利用するDNSサーバの実装によらず同じ手法を使えます。

しかし、TLSではキャプチャしたパケットは暗号化されています。いまはPerfect Forward Secrecy(PFS)が当たり前になっていて、サーバの秘密鍵があっても復号できません。これまで情報取得に使っていたツールがまったく使えなくなるということです。内々の実験ならともかく、サービスとして広く使っていただくためにはこれができなければ提供を断念せざるを得ないぐらい必須の機能です。そのためパケットキャプチャに頼らずDNSサーバ自体から得られる情報を用いてこれまで同様の統計情報を取得する基盤を作り直すことになり、ようやくサービス開始できるところにこぎ着けました。

2.3.3 HTTPの壁

実はHTTPの壁はそんなに高くありません。

DoHでは、TLSの後で更にDNSメッセージをHTTPメッセージにカプセル化する必要があります。DNSサーバ自身にHTTPを喋らせようとするのならばかなり苦労するでしょうが、一般的なHTTPサーバで問い合わせを受け、メッセージ形式を変換して背後のDNSサーバと通信するような2段構成であれば、バックエンドがDNSサーバであること以外はどこにでもある当たり前のWebアプリケーションでしかありません。

従来のUDPではなくTCPでありTLSである以上、レイテンシその他のパフォーマンスの問題から逃れられないのは明らかです。しかし、これまでの蓄積がなく手探りで行わなければならないDNSのレイヤーではなく、これまでに十分な実績のあるHTTPのレイヤーで面倒な部分の大半を解決できるため、それほど大きくつまずくことはありません。

2.3.4 壁は乗り越えられたのか

サービスを開始してから本稿執筆時点で約4ヵ月が経っています。

プレスリリース時には国内初のDoT/DoHサービスとしてそこそこ話題になり、Android用DoHクライアントであるIntra[*3]の設定UIでは、選択肢からIIJ Public DNSサービスを選ぶだけで使えるようにもなりました(選択肢に入れてくれとこちらからお願いしたわけではないのですが)。

*3 Intra(https://play.google.com/store/apps/details?id=app.intra&hl=ja)。

順調にスタートを切ってまったく問題なし、と胸を張りたいところですが、残念ながらそうなってはいません。

前述のとおり、UnboundにはもともとTLSセッション再開機能がなく、それを我々の手で実装したのですが、Android9のDoTはTLSセッションの有効期間が非常に短いようで、せっかく実装したセッション再開が有効に機能しないケースが多いことが分かっています。AndroidスマートフォンでWebページを読み込み、その後別のページを見に行こうとリンクを辿ると、そのときには前回の名前解決の際に確立されたTLSセッションが既にタイムアウトしており、ハンドシェイクを初めからやり直さなければならない、という事象が頻発するのです。ネットワークが混雑しているようなケースではこのハンドシェイクに失敗することも多々あり、結果として名前解決ができない、ネットが使えない、という現象がたびたび起きてしまうのです。

しかも、現在（執筆時点で）開発中ベータのAndroid10では、現行のAndroid9よりも更にパフォーマンスが悪化してしまうようです。

これ以外については特に大きな問題は起きていません。従来のDNSに比べてレイテンシが悪化するはずですが、体感的には問題ないようでお叱りを受けることもありません。

DoT/DoHはまだ新しい技術であり、根幹の仕様がやっと固まったばかりで、周辺仕様はこれから決める、という部分が多く残っています（例えば、現時点ではDoT/DoHサーバは手作業で設定するしかなく、ネットワーク管理者が設定を配布して自動で適用させる、ということができません）。

今後は問題点の改善に向けて引き続き調査・検討を進めると共に、将来標準化される新しい仕様を実装して最新の技術を安心して使っていただけるよう努め、また、サービスの運用で得られた知見をコミュニティに広く還元していく予定です。

2.4 パブリックDNSとDoT/DoH

最後に、IIJ Public DNSサービス以外の動きについて概観します。

本来キャッシュDNSサーバは組織内部のユーザに提供するもので、外部に公開する性質のものではありませんが、公開してもそれほど害はないという認識だったため、アクセス制限されていないもの（オープンリゾルバ）が大半でした。しかし、DNS ampという攻撃手法が発見され、DDoS攻撃の踏み台として広く利用されるようになったため、2010年頃からは必要な範囲からのアクセスだけに制限するのが常識になっています。

その一方で、接続元アドレスによる制限ではなくレート制限などの方法で攻撃の対策をした上で、幅広いユーザに使ってもらおうというサービスが出てきました。その先駆者となったのはOpenDNS[*4]ですが、その後に登場したGoogle Public DNS[*5]が定着して以降は、このような意図してオープンリゾルバにしているサービスを「パブリックDNS」と呼ぶのが通例になっています。

DoTのRFCが標準化されたのは2016年です。翌2017年11月にサービス開始したパブリックDNSであるQuad9[*6]、2018年4月開始のCloudflare[*7]は当初から、Googleも2019年1月からDoTに対応しています。

DoHが正式にRFCになったのは2018年10月ですが、ドラフトをベースにした実装が先行し、Cloudflareでは2018年4月の開始当初から、Quad9もRFC8484が出る2週間前にDoHに対応し、遅れてGoogleも2019年6月に対応しました。

クライアント側でもAndroidがOSとして2018年8月からDoTに、10月にアプリとしてDoHクライアントIntraがリリースされています。WebブラウザではFirefoxが2018年8月にDoHに対応、Chromeは執筆時点では開発版でのみの対応ですが、本稿が世に出る頃には正式版でも利用できるようになっているかもしれません。

＊4　OpenDNS（https://www.opendns.com/）。
＊5　Google Public DNS（https://developers.google.com/speed/public-dns/）。
＊6　Quad9（https://www.quad9.net/）。
＊7　Cloudflare（https://developers.cloudflare.com/1.1.1.1/）。

このように、DoT/DoHへの対応は急速に進んでいますが、その一方で懸念もないわけではありません。

Firefoxではゆくゆくは名前解決をデフォルトでDoHにしたい、すなわちユーザが特に設定しなければFirefoxが選定したパブリックDNSサービスを自動で使わせる意向であると伝えられています[*8]。しかし、パブリックDNSはパブリックであるが故に、イントラネットなどプライベートな名前空間の解決ができません。また、DNSを利用したペアレンタルコントロールサービスなどを利用している場合、OSの設定とは異なるDNSサーバをブラウザに勝手に使われてしまうと制御できなくなってしまいます。こういった点から、デフォルトでDoHの設定を「押しつける」ことの是非については議論があります。

Google Public DNSはDoT/DoHに対応しており、これを使うことでユーザとGoogleの間の機密性が保証されることになります。その一方で、GoogleはEDNS0 Client Subnet (ECS; RFC7871) にも対応しています。ECSはキャッシュサーバに問い合わせてきたクライアントの属するネットワークの情報をキャッシュサーバから権威サーバに伝えることで、コンテンツ配信のトラフィックマネジメントに役立てようというものですが、ECSが使われるキャッシュサーバと権威サーバの間は常に従来のDNSが使われるということに注意しなければなりません。ユーザがDoT/DoHを使っていればGoogleまでの間の経路上では盗聴されませんが、Googleと権威サーバの間の経路上は機密性のない従来のDNSが使われるため盗聴が可能で、ここに含まれるECSの情報からユーザのプライバシーが漏れかねないのです。

DNSトランスポートが暗号化されていくという流れ自体はもはや確定的であり、これを押しとどめることは到底できそうもありません。しかし、すべてを是とするのではなく、ほんとうにそれは正しいのかひとつひとつ検証していくこともまた、新たにパブリックDNSサービスを始めた我々の重要な責務といえるでしょう。

執筆者:
山口 崇徳（やまぐち たかのり）
IIJ アプリケーションサポート部アプリケーションサポート課。DNSサービスのサポート等に従事。

*8 Firefox Nightly News、"What's next in making Encrypted DNS-over-HTTPS the Default" (https://blog.mozilla.org/futurereleases/2019/09/06/whats-next-in-making-dns-over-https-the-default)。