HTTPS リソースレコードへの期待

アカマイ・テクノロジーズ合同会社
ソリューションズ・エンジニア
松本 陽一

このプレゼンテーションにおいてなされる記述は作成者個人の見解を示すものであり、アカマイ・テクノロジーズの見解を示すものではありません。提供される情報は作成時点において正確なものであると考えておりますが、当該情報についてなんら表明又は保証を行いません。

本日の流れ

- 新しい DNS リソース・レコード・タイプ である HTTPS の概要
- HTTPS RR のもたらすもの（効果）
- CDN や大規模配信等における課題解決への期待

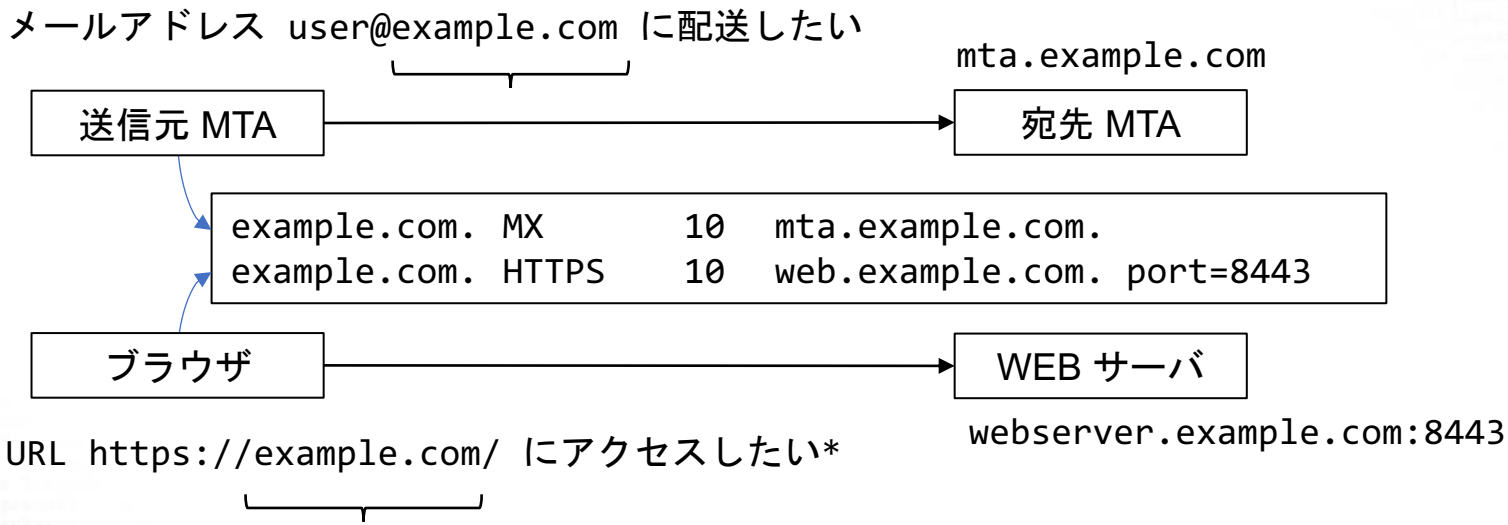
網羅的な解説よりも、使いたくなる場面を中心に

HTTPS リソースレコードの概要

- 新しい DNS リソースレコードタイプ
  draft-ietf-dnsop-svcb-https
  "Service binding and parameter specification via the DNS (DNS SVCB and Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)"
  SVCB (64) と HTTPS (65) を定義。
- ブラウザが URL にアクセスする際、URL ホスト名*に関して HTTPS タイプのクエリを行う。
  並行して従来通り URL ホスト名に関する AAAA と A もクエリするが HTTPS クエリの応答を優先して扱う(3 つのクエリが飛ぶ)
- iOS14 リリース (2020/9) 以来、実際に多く観測されている。Firefox や Chrome にも条件付きで実装されている。

* 今日のセッションでは https://www.example.com/memu.html の「www.example.com」の部分をこう呼びます

HTTPS RR (ServiceMode)
メール配送における MX と似ている点、違う点



- MTA は配送先の MTA を メールアドレスのホスト部に対する MX レコードで見つける
- ブラウザがアクセス先の WEB サーバを URL のホスト部に対する HTTPS レコードで見つける
- HTTPS ではそれに加えて「どのように」アクセスするかをパラメータ指定できる (この例では、ポート番号)

* https://example.com:4431/ の場合は _4431._https.example.com. (QTYPE=HTTPS) をクエリ

ServiceMode のフォーマット

```
Owner Name                RRType SvcPriority Target                SvcParam(s)
www.example.com.          HTTPS         10 host1.example.net. alpn="h3,h2"
                          HTTPS         10 host2.example.net. alpn="h3,h2"
                          HTTPS         20 host3.example.net. alpn="h2" port="8443"
```

- 優先度が同じ host1、host2 からランダムに選択し、HTTP/3 または HTTP/2 で接続 (alpn: Application Layer Protocol Name)
- host1、host2 もだめな場合、優先度が低い host3 のポート 8443 に HTTP/2 で接続できる (alpn="http/1.1" がデフォルトであり TCP 上の TLS でネゴするので alpn="h2" はなくても同じはず)

AliasMode のフォーマット (SvcPriority = 0 だと AliasMode)

```
Owner Name          RRType SvcPriority Target
example.com.        HTTPS         0  www.example.com.
```

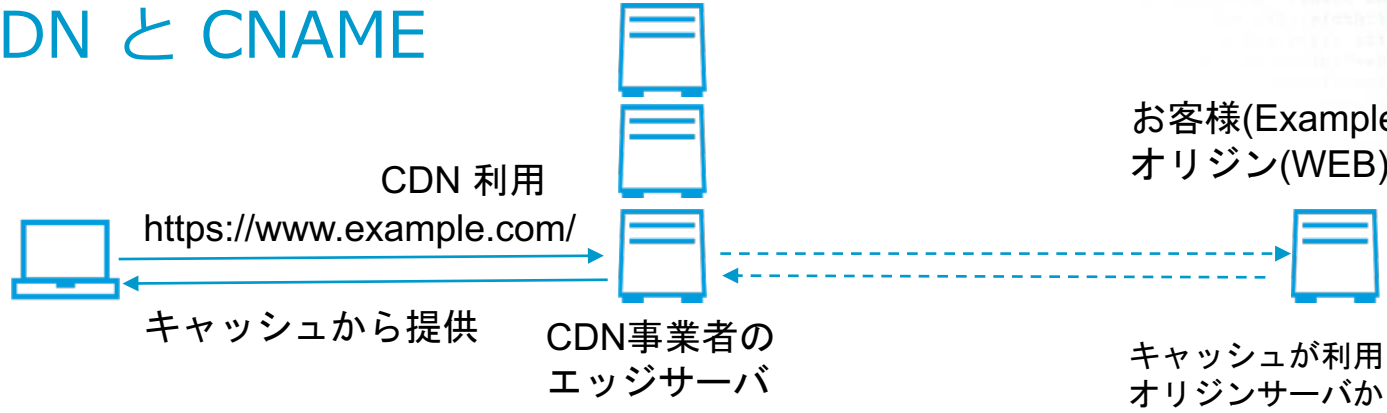
単純な「別名」であり、SvcParam はつけられない (CNAME と似ている)
→ クライアントは次に www.example.com. の HTTPS RR を検索する (↔ ServiceMode)

HTTPS RR のもたらすもの（効果）

- CNAME と異なりゾーン頂点に設定できる別名
- アクセス分散とフォールバック
- 平文 HTTP による接続を防止 (http: → https:)
- はじめから HTTP/3 (QUIC) による接続を可能に (alpn)
- ECH (Encrypted Client Hello) の公開鍵等の取得 (ech)
- デフォルト以外のポート番号の指定 (port)
- その他の新しいパラメータによる拡張 (keyXXX)

など

CDN と CNAME



- ブラウザがアクセスするのは CDN のエッジサーバ*
- www.example.com はお客様の管理するドメイン名でありながら、CDN 事業者の管理する IP アドレスに解決されなければいけない
- 多くの CDN ではお客様の権威ネームサーバで CNAME を設定してもらう。
  例) `www.example.com. 1800 IN CNAME www.example.com.edgekey.net.`

edgekey.net. はアカマイが管理するゾーン。www.example.com.edgekey.net はアカマイのエッジサーバ

* CDN の役割はキャッシュだけでなくセキュリティ、画像最適化、コード実行などに拡大しており、CDN 事業者はキャッシュ・サーバとは呼ばない傾向

CNAME の課題

CNAME は他のリソースレコードと共存できない、とくに：

- CNAME はゾーン頂点に設定できない
  https://www.example.com/ は可能だが、 https://exmple.com/ は不可
- CNAME は複数設定できない

現状の解決方法の例

- CDN 事業者の提供する権威 DNS サービスを利用
  エッジサーバの IP アドレスを返す機能を提供
  (例: アカマイの Edge DNS における Zone Apex Mapping)
- DNS GSLB による分散 – 権威ネームサーバで問い合わせ (ソース IP アドレス=フルリゾルバ) 毎に動的に異なる CNAME 応答
  (例: アカマイの GTM - Global Traffic Management)

HTTPS RR は CNAME の課題を解決できるのか

- HTTPS RR は CNAME と異なりゾーン頂点に設定できる

しかし…

- HTTPS RR に対応しないクライアントには参照されない
  →
  たとえ 95% のクライアントが HTTPS RR に対応しても残り 5% のクライアントに A/AAAA を返さないといけない
  → 現行の対応(権威ネームサーバでの特殊な対応)が必要

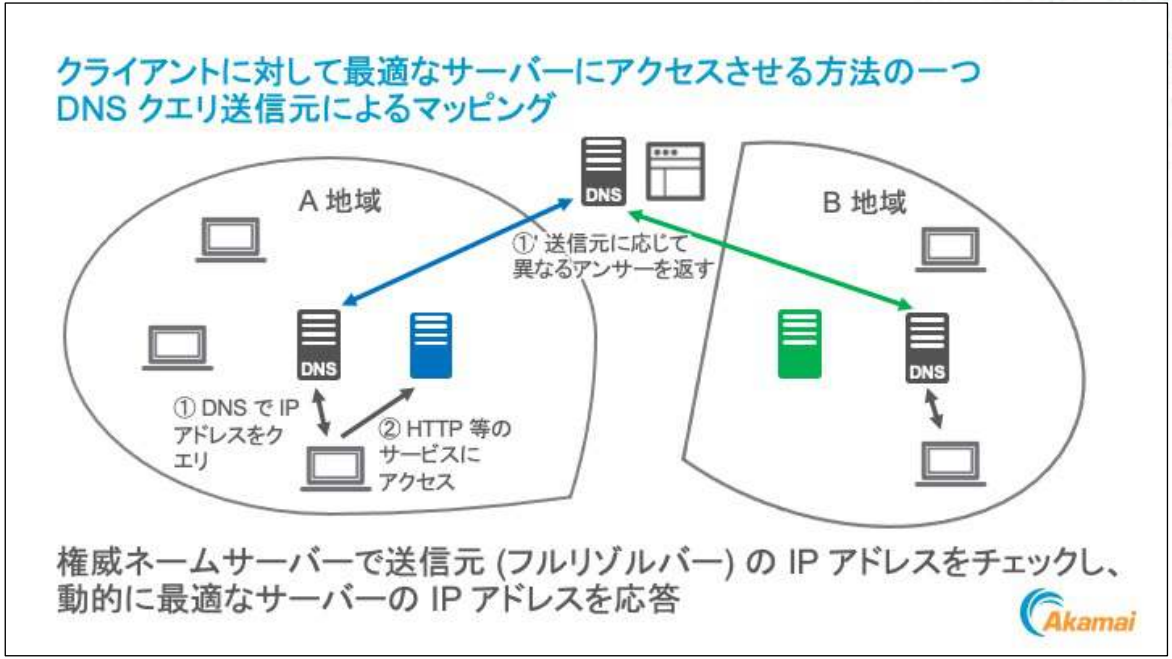
他の目的での HTTPS RR の利用が先の可能性

DNS によるグローバル負荷分散 (GSLB)

大規模配信では多数のサーバを分散配置し、適切なサーバにクライアントのアクセスを導く必要がある。

- ネットワーク上近いだけでなく
- 輻輳していない/させない
- 過負荷でない/にしない
- 障害中でない

CDN の権威ネームサーバで、クエリの送信元 (フルリゾルバー) に応じて最適なサーバを応答



DNS Summer Day 2019 の発表資料

https://dnsops.jp/event/20190628/DNS_Summer_Day_2019-EDNS_Client_Subnet-Matsumoto.pdf

DNS GSLB の課題

- 同じフルリゾルバを使用している全てのユーザに (TTL満了まで)同じ応答が返される
  → 多数のユーザがいる場合に細やかなコントロールができない
- 複数の A / AAAA が応答された場合のフルリゾルバの動作も、フルリゾルバから複数の A / AAAA が応答された場合のクライアントも、実装や設定次第
- サーバの障害を検出して DNS GSLB での応答を変更してもフルリゾルバのキャッシュの TTL の間は反映されない
- 局所的なネットワークの問題は検出できない場合も

HTTPS RR (ServiceMode) によるロードバランス

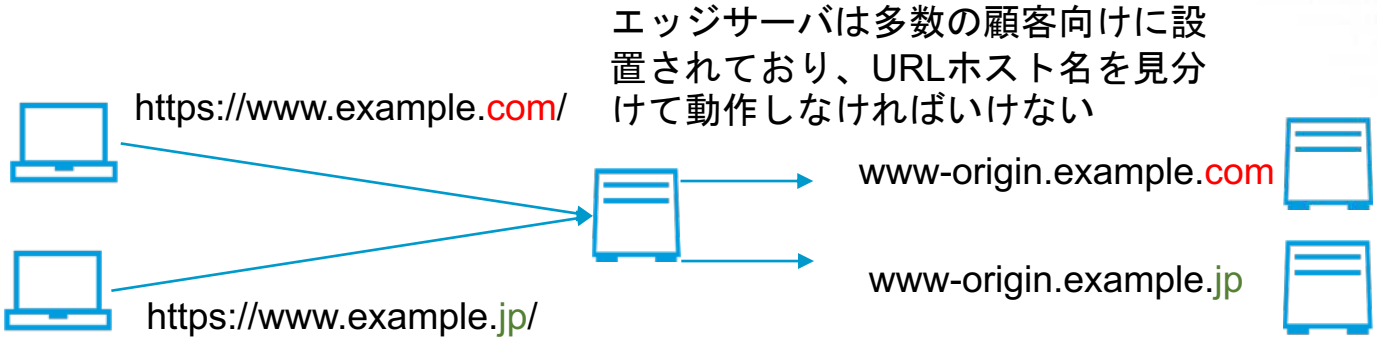
- 複数の Target が書け、ランダムに選ばれることが規定される
- 各々の A/AAAA の TTL や SvcParam は異なるものに設定できる
- 割合を変えることも…

```
www.example.com.    HTTPS 1 name1.example.com. alpn="h2"
                    HTTPS 1 name2.example.com. alpn="h2"
                    HTTPS 1 name3.example.com. alpn="h2"
name1.example.com. A     192.0.2.1
name2.example.com. A     192.0.2.1
name3.example.com. A     192.0.2.2
```

2:1 に按分

- サーバの障害時にクライアントが優先度の低いサーバにフォールバックしてくれる
  → GSLB においては、例えば東京のバックアップを大阪に、大阪のバックアップを東京にといったことができる

共用サーバにおける URL ホスト名の見分け方



多数の顧客の多数の URLホスト名をサポートするため複数のサーバ証明書を持つが、コネクションを確立する段階で見分ける必要がある (Host: ヘッダに頼れない)

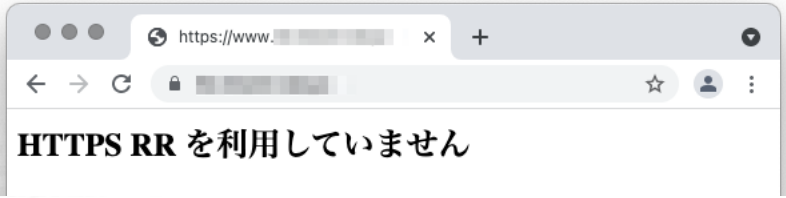
- IPアドレスを複数もつ方法
- SNI (Server Name Indication) TLS 拡張を利用 – 接続時にクライアントから送られる Client Hello メッセージに載るサーバ名 (URL ホスト名) を利用

通常のブラウザは SNI に対応しており後者が原則

HTTPS RR に対応しているブラウザと対応していないブラウザで異なる IP アドレスにアクセスさせる

[実験] URL ホスト名に対して HTTPS RR の返すエンドポイントの IP アドレスと A/AAAA の返す IP アドレスを違うものにする → 対応していれば HTTPS RR が優先される。

```
www.example.com.      HTTPS 1 server2.example.com. alpn="http/1.1"
www.example.com.      A     192.0.2.1 ← 対応していないブラウザがアクセス
server2.example.com.  A     192.0.2.2 ← 対応しているブラウザがアクセス
```



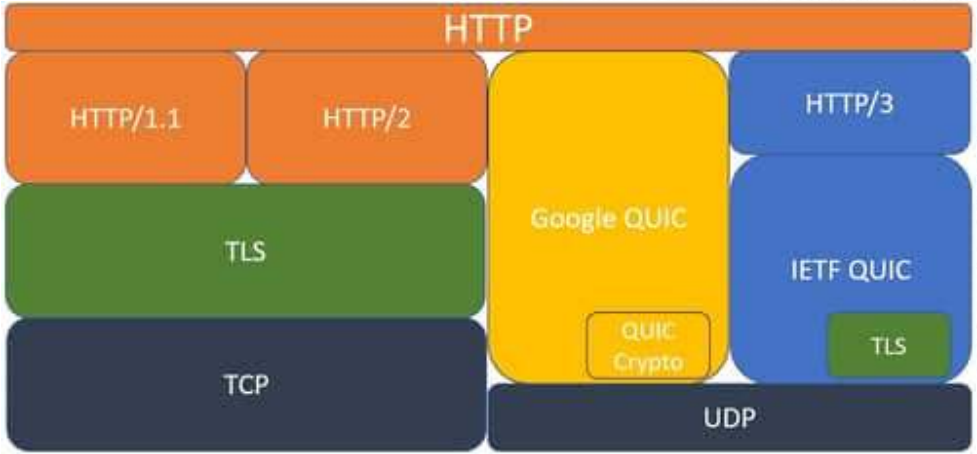
Chrome 95

Safari 15.1

macOS Monterey (12.0.1)

[応用] 新しいブラウザはグローバルに分散させつつ (SNI に対応しない等)、 古いブラウザは IP アドレス方式で限られたサーバにアクセスさせる。後方互換性とスケーラビリティ両立の課題を解決

はじめから HTTP/3 (QUIC) 接続



[現状]

- HTTP/1.1 と HTTP/2 (H2) は？
  → 接続時に
  TLS Client/Server Hello (alpn)で決まる
- HTTP/3 (H3) は？
  → 最初は H2 (TLS/TCP) で接続
  Alt-Svc ヘッダで H3 サポートを知ると、
  H3 (QUIC/UDP) で接続可能

[HTTPS RR を使うと]

- はじめから H3 で接続可能
  → ブラウザが H3 と HTTPS RR の両方に対応し、
  サーバ側で H3 に対応するならHTTPS RR は書きたい

ステータス	メソッド	ドメイン	ファイル	プロトコル
200	GET	www.google.com	/	HTTP/2
200	GET	www.google.com	m=cdos,c	HTTP/3
200	GET	www.google.com	googlelog	HTTP/3
200	GET	www.gstatic.com	rs=AA2YrT	HTTP/2
200	GET	www.gstatic.com	rs=AA2YrT	HTTP/3

上図は Mike Bishop のブログより
https://www.akamai.com/blog/performance/http3-and-quic-past-present-and-future

ECH (Encrypted Client Hello)

- TLS では通信の中身を暗号化するが、SNI は平文で流れるので中間者や盗聴者にアクセス先のホスト名が分かってしまう。
- ECH (Encrypted Client Hello) – SNI を含めた Client Hello の暗号化
  クライアントが公開鍵を含む情報 (ECHConfig) を取得する主な方法として HTTPS RR を採用 draft-ietf-tls-esni
  (TXT レコード等に載せる方法では秘密鍵を共有しない複数のサーバに分散しづらいが、HTTPS RR なら個別に設定できる)
- SNI が暗号化されても IP アドレスが分かる点に変わりはない

今のところは不確定要素が多いようだが、今後も要注視

その他の新しいパラメータキーの提案

- dohuri (32768)
  "Adaptive DNS Resolver Discovery"
  draft-pauly-add-resolver-discovery
  だだし expired
- odohconfig (32769)
  draft-pauly-dprive-oblivious-doh-04
  "Oblivious DNS Over HTTPS"
  ただし draft 05 で削除された

[参考]

- dohpath (7)
  draft-ietf-add-svcb-dns
  "Service Binding Mapping for DNS Servers"
  HTTPS RR ではなく SVCB のパラメータ

クライアントの HTTPS RR サポート状況 (11月上旬現在の推定)

実験的にはかなり前から実装されていたりするが、一般的には…

- iOS: iOS14 (2020 年 9 月) から (http を https に書き換える効果はない)
- macOS Safari: Big Sur 以降の Safari のみ (同上)
- Firefox: 92 (2021 年 9 月) – DoH のときのみ
  http を https に書き換える効果は確認できるが..。
- Chrome: HTTPS RR クエリを行うか、DNS 暗号化を条件とするか、
  http→https を行うか等のスイッチがあり Google からコントロールして実験中 (未知のタイプとして TYPE65521 をクエリする実験も行ってきた)
  11月上旬現在 96 Beta で 50% が HTTPS RR クエリを行うらしい
  (HTTPS RR が応答されしても効果がない場合もあるようにみられる)
  96 Stable 化 (2021 年 11 月 16 日?) 以降、徐々に展開するとのこと

まとめ

CDN ではお客様の WEB サイトのかわりに HTTPS を終端してサービスを提供するため、新しいテクノロジー / プロトコルに対応する必要

HTTPS RR は、HTTPS とその利用形態のあらゆる進化に関わりを持っており、ブラウザの対応状況とともに、利用が広がっていく大きな可能性を持っている

DNS と HTTP のセキュリティや、プライバシーの議論の 1 ピースとして今後も要注目


Akamai