JP AAWG Japan Anti-Abuse Working Group

OX Stay Open.

暗号化DNSフォワーダを作ってみた

OX Dovecot 株式会社

利波 健二

インターネットの世界は暗号化があたりまえ

インターネットで利用するプロトコルはどんどん暗号化

- telnet → ssh(sftp, scp)
- HTTPS
  - あらゆるWebベースのサービス
- FTPS
- SMTPS、IMAPS、POP3S
- LDAPS

でもDNSは…

- 特に暗号化は必要ないのか？
  - DNSクエリには機密情報はない?
  - プロバイダのネットワークがセキュアなら盗み見されないっしょ

JP AAWG Japan Anti-Abuse Working Group

OX Stay Open.

もし、知られたら？

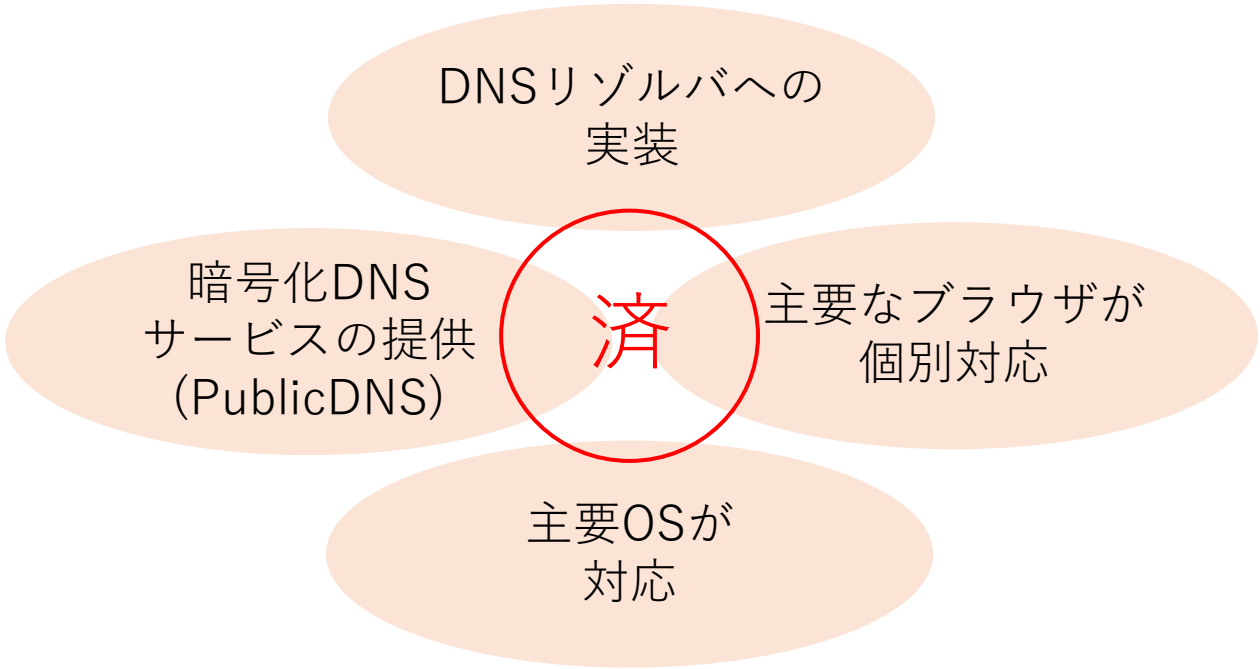
- DNSクエリが見えたら…

- 使っているサービスがわかる
  - 銀行やカード会社
  - ECサイト
  - SNS
  - あまり知られたくないサイト(笑)

- ピンポイントにログインを試したり
- フィッシングメールの精度が上がるかも
  - 現状は、使ってもいないサービスの怪しいメールの大量生産 (→)
  - 自分が使っているサイトの通知だけになったら慎重になるかも

環境は整いつつある

- 暗号化DNSの実装
  - 主要なOSSは実装済
- DNSサービス
  - 大手によるPublicDNSとして公開
  - 個別のプロバイダのサービスは…
- DNSクライアント
  - 主要なブラウザは対応
  - OSも対応が進み、内部DNSが条件によって利用

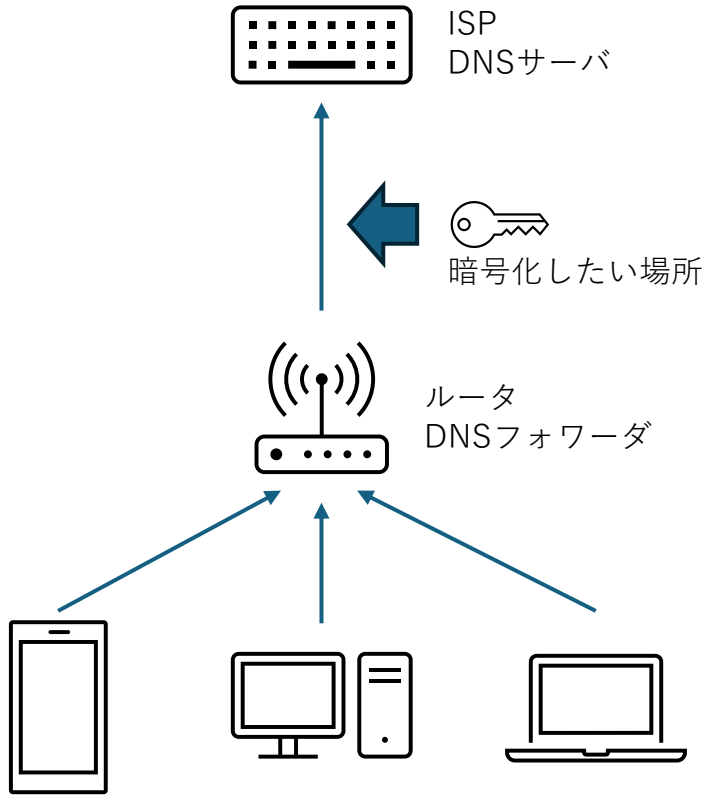


現在のDNSの環境

- 家でPCやスマホを使う場合
  - 端末はルータのDNSフォワーダへ問い合わせ
  - ルータはプロバイダのDNSへ問い合わせ

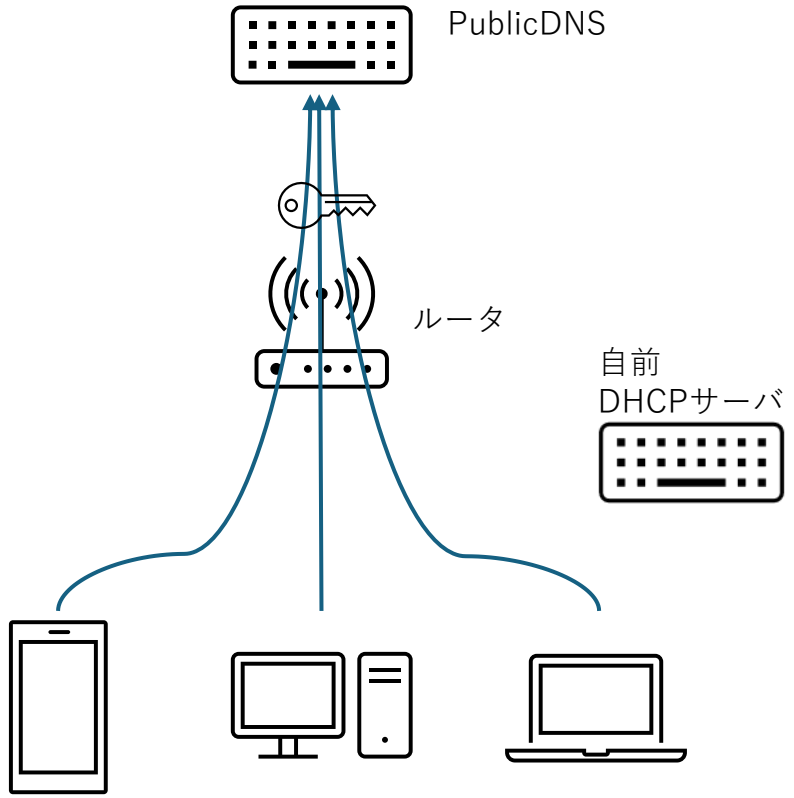
- 暗号化で頑張ってほしいのは
  - ルータのDNSフォワーダ
  - プロバイダのDNSリゾルバ

※最近のASUSやtp-linkのルータは使用できるものがありそう



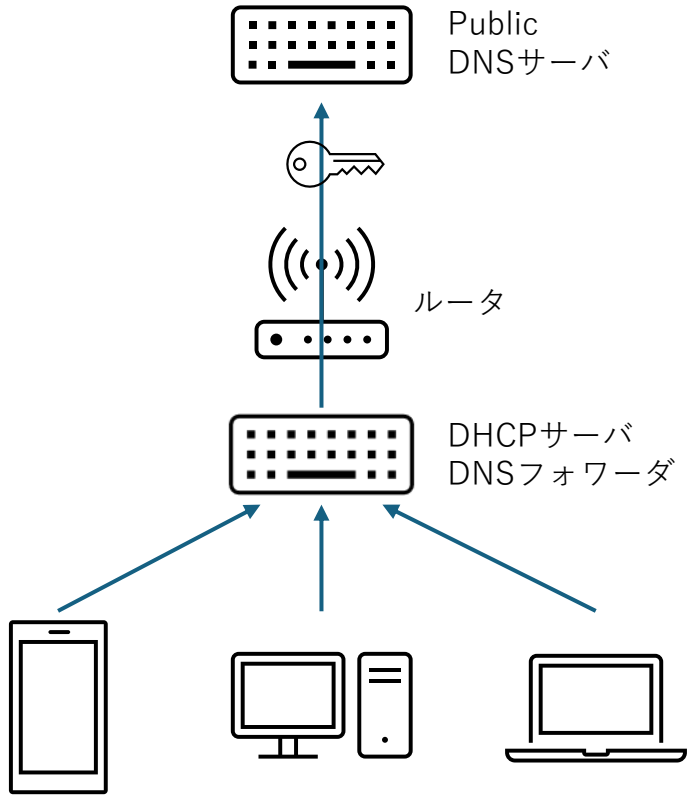
我が家のルータは…

- 初心者向けゆえ、設定項目がない
  - DHCP機能を有効化すると、DNSサーバはLAN側IP固定
  - ルータが問い合わせするDNSサーバは固定
- できることといえば
  - ルータのDHCPサーバを無効化
  - DHCPサーバを別途用意してDNSサーバをpiblicDNSに変更
  - 各クライアントがDoT、DoHを利用するように設定
- 個別に有効化したり面倒だなー



自前のDNSフォワーダでDNSクエリ暗号化

- DNSフォワーダを自前で作ってしまおう
  - クライアントの設定はほったらかし
  - 未対応のクライアントでも暗号化クエリへ変換



dnsdist

- dnsdistはDNS用ロードバランサ
  - Proxyのように動作し、受けたクエリを(複数の)指定したサーバへ転送
  - 受信側、送信側 ともに暗号化DNSクエリへ対応

- 今回の目的には使えそう

- DDoS対応などの話については https://dnsdist.org をご確認ください

dnsdistのインストール

- インストール

```
# apt-get install dnsdist
```

Ubuntu 24.04 LTSではdnsdist 1.8.3 がインストールされる

最新の1.9系を利用したい場合は、PowerDNS repositoriesからダウンロードすることで利用可能

まずは単純なフォワーダ設定

- /etc/dnsdist/dnsdist.confを編集

```
-- Listen IPaddress
addLocal("192.168.1.71")

-- DNS Server IPaddress
newServer("8.8.8.8")
```

これで、dnsdistで受け取ったDNSクエリを 8.8.8.8 (google Public DNS) へ問い合わせしてくれます

まずは単純なフォワーダ設定

- tcpdumpで確認

No.	Time	Source	Destination	Protocol	Lengtl	Info
13	6.001955	192.168.1.71	8.8.8.8	DNS	78	Standard query 0x57de A a.root-servers.net
14	6.026154	8.8.8.8	192.168.1.71	DNS	94	Standard query response 0x57de A a.root-servers.net A 198.41.0.4
15	6.378103	192.168.1.81	192.168.1.71	DNS	99	Standard query 0x2ce6 A www.google.co.jp OPT
16	6.378262	192.168.1.71	8.8.8.8	DNS	99	Standard query 0x0100 A www.google.co.jp OPT
17	6.423322	8.8.8.8	192.168.1.71	DNS	103	Standard query response 0x0100 A www.google.co.jp A 172.217.161.195 OPT
18	6.423941	192.168.1.71	192.168.1.81	DNS	103	Standard query response 0x2ce6 A www.google.co.jp A 172.217.161.195 OPT
19	7.002273	192.168.1.71	8.8.8.8	DNS	78	Standard query 0xe06f A a.root-servers.net
20	7.024918	8.8.8.8	192.168.1.71	DNS	94	Standard query response 0xe06f A a.root-servers.net A 198.41.0.4

届いたクエリを8.8.8.8へフォワードしてくれています。

ですが、前後に同じ問い合わせが繰り返し発生。dnsdistがロードバランサであるため、ヘルスチェックをしています(default)。公開サーバにこれは迷惑すぎる、ということで対策します

lazyヘルスチェック

- /etc/dnsdist/dnsdist.confを編集

```
-- Listen IPaddress
addLocal("192.168.1.71")

-- DNS Server IPaddress
newServer({address="8.8.8.8", healthCheckMode='lazy', checkInterval=1, lazyHealthCheckFailedInterval=30, rise=2,
maxCheckFailures=3, lazyHealthCheckThreshold=30, lazyHealthCheckSampleSize=100, lazyHealthCheckMinSampleCount=10,
lazyHealthCheckMode='TimeoutOnly'})
```

デフォルトは、DNS Serverの稼働状況を確認するためのヘルスチェックだが、lazy modeを指定することで、DNSの問い合わせが失敗したときに復活待ちのヘルスチェックを行います。
復活を確認するとヘルスチェックが停止します。

lazyヘルスチェック

- tcpdumpで確認

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.81	192.168.1.71	DNS	99	Standard query 0xc4e1 A www.google.co.jp OPT
2	0.000160	192.168.1.71	8.8.8.8	DNS	99	Standard query 0x0100 A www.google.co.jp OPT
3	0.018542	8.8.8.8	192.168.1.71	DNS	103	Standard query response 0x0100 A www.google.co.jp A 142.250.206.227 OPT
4	0.018726	192.168.1.71	192.168.1.81	DNS	103	Standard query response 0xc4e1 A www.google.co.jp A 142.250.206.227 OPT
5	12.143943	192.168.1.81	192.168.1.71	DNS	98	Standard query 0x9274 A www.yahoo.co.jp OPT
6	12.144093	192.168.1.71	8.8.8.8	DNS	98	Standard query 0x0200 A www.yahoo.co.jp OPT
7	12.157563	8.8.8.8	192.168.1.71	DNS	130	Standard query response 0x0200 A www.yahoo.co.jp CNAME edge12.g.yimg.jp A 1
8	12.157709	192.168.1.71	192.168.1.81	DNS	130	Standard query response 0x9274 A www.yahoo.co.jp CNAME edge12.g.yimg.jp A 1

Googleとyahooの問い合わせの間にヘルスチェック通信がない

1秒ごとのヘルスチェッククエリがなくなりました

DNS over TLS導入(クエリの暗号化)

- /etc/dnsdist/dnsdist.confを編集

```
-- Listen IPaddress
addLocal("192.168.1.71")

-- DNS Server IPaddress
newServer({address="8.8.8.8:853", healthCheckMode='lazy', checkInterval=1, lazyHealthCheckFailedInterval=30,
rise=2, maxCheckFailures=3, lazyHealthCheckThreshold=30, lazyHealthCheckSampleSize=100,
lazyHealthCheckMinSampleCount=10, lazyHealthCheckMode='TimeoutOnly', tls="openssl"})
```

ポートを指定すれば完了です。

DNS over TLS導入(クエリの暗号化)

- tcpdumpで確認

	No.	Time	Source	Destination	Protocol	Length	Info
①	1	0.000000	192.168.1.81	192.168.1.71	DNS	99	Standard query 0x8181 A www.google.co.jp OPT
	2	0.000344	192.168.1.71	8.8.8.8	TCP	74	47930 → 853 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=656129783 TSecr=0 WS=128
	3	0.011935	8.8.8.8	192.168.1.71	TCP	74	853 → 47930 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM TSval=1880149024 TS
	4	0.011957	192.168.1.71	8.8.8.8	TCP	66	47930 → 853 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=656129795 TSecr=1880149024
	5	0.012719	192.168.1.71	8.8.8.8	TLSv1.3	671	Client Hello
	6	0.024548	8.8.8.8	192.168.1.71	TCP	66	853 → 47930 [ACK] Seq=1 Ack=606 Win=65024 Len=0 TSval=1880149038 TSecr=656129796
	7	0.062547	8.8.8.8	192.168.1.71	TLSv1.3	285	Server Hello, Change Cipher Spec, Application Data
	8	0.062569	192.168.1.71	8.8.8.8	TCP	66	47930 → 853 [ACK] Seq=606 Ack=220 Win=64128 Len=0 TSval=656129846 TSecr=1880149075
③	9	0.063228	192.168.1.71	8.8.8.8	TLSv1.3	146	Change Cipher Spec, Application Data
	…	0.063820	192.168.1.71	8.8.8.8	TLSv1.3	147	Application Data
	…	0.073997	8.8.8.8	192.168.1.71	TCP	66	853 → 47930 [ACK] Seq=220 Ack=767 Win=65024 Len=0 TSval=1880149087 TSecr=656129846
	…	0.075616	8.8.8.8	192.168.1.71	TLSv1.3	695	Application Data, Application Data
②	…	0.075879	192.168.1.71	192.168.1.81	DNS	103	Standard query response 0x8181 A www.google.co.jp A 142.250.76.131 OPT

クライアントからdnsdist間の内容(①、②)は見えていますが、PublicDNSとのやり取り(③)は見えなくなっています。

dnsdistでレコードのキャッシュ

- /etc/dnsdist/dnsdist.confを編集

```
-- Listen IPaddress
addLocal("192.168.1.71")

-- DNS Server IPaddress
newServer({address="8.8.8.8:853", healthCheckMode='lazy', checkInterval=1, lazyHealthCheckFailedInterval=30,
rise=2, maxCheckFailures=3, lazyHealthCheckThreshold=30, lazyHealthCheckSampleSize=100,
lazyHealthCheckMinSampleCount=10, lazyHealthCheckMode='TimeoutOnly', tls="openssl"})

-- DNS cache
pc = newPacketCache(10000)
getPool(""):setCache(pc)
```

newServerで指定したサーバは、特に指定しなければデフォルトのサーバプールに組み込まれます。
newPacketCacheでキャッシュ領域を確保し、getPoolで割り当てるとキャッシュが効くようになります。

dnsdistでレコードのキャッシュ

- tcpdumpで確認

	No.	Time	Source	Destination	Protocol	Lengtl	Info
①	1	0.000000	192.168.1.81	192.168.1.71	DNS	99	Standard query 0xf6bb A www.google.co.jp OPT
	2	0.000315	192.168.1.71	8.8.8.8	TCP	74	44626 → 853 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=664235460 TSecr=0 WS=128
	3	0.014876	8.8.8.8	192.168.1.71	TCP	74	853 → 44626 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM TSval=3800508708 TSecr=664235460 WS=256
	4	0.014906	192.168.1.71	8.8.8.8	TCP	66	44626 → 853 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=664235475 TSecr=3800508708
	5	0.015783	192.168.1.71	8.8.8.8	TLSv1.3	671	Client Hello
	6	0.029853	8.8.8.8	192.168.1.71	TCP	66	853 → 44626 [ACK] Seq=1 Ack=606 Win=65024 Len=0 TSval=3800508724 TSecr=664235476
	7	0.069686	8.8.8.8	192.168.1.71	TLSv1.3	285	Server Hello, Change Cipher Spec, Application Data
	8	0.069729	192.168.1.71	8.8.8.8	TCP	66	44626 → 853 [ACK] Seq=606 Ack=220 Win=64128 Len=0 TSval=664235530 TSecr=3800508763
	9	0.071648	192.168.1.71	8.8.8.8	TLSv1.3	146	Change Cipher Spec, Application Data
	…	0.073180	192.168.1.71	8.8.8.8	TLSv1.3	147	Application Data
	…	0.088347	8.8.8.8	192.168.1.71	TCP	66	853 → 44626 [ACK] Seq=220 Ack=767 Win=65024 Len=0 TSval=3800508783 TSecr=664235532
	…	0.094681	8.8.8.8	192.168.1.71	TLSv1.3	695	Application Data, Application Data
	…	0.094949	192.168.1.71	192.168.1.81	DNS	103	Standard query response 0xf6bb A www.google.co.jp A 142.250.76.131 OPT
	…	0.135672	192.168.1.71	8.8.8.8	TCP	66	44626 → 853 [ACK] Seq=767 Ack=849 Win=63616 Len=0 TSval=664235596 TSecr=3800508789
②	…	8.781689	192.168.1.81	192.168.1.71	DNS	99	Standard query 0x277c A www.google.co.jp OPT
	…	8.781856	192.168.1.71	192.168.1.81	DNS	103	Standard query response 0x277c A www.google.co.jp A 142.250.76.131 OPT

初回の問い合わせ(①)はPublicDNSと通信していますが、2回目 (②)はdnsdistが即答。

仕上げ

- /etc/dnsdist/dnsdist.confを編集

```
-- Listen IPaddress
addLocal("192.168.1.71")

-- DNS Server IPaddress
newServer({address="103.2.57.5:853", healthCheckMode='lazy', checkInterval=1, lazyHealthCheckFailedInterval=30, rise=2, maxCheckFailures=3,
lazyHealthCheckThreshold=30, lazyHealthCheckSampleSize=100, lazyHealthCheckMinSampleCount=10, lazyHealthCheckMode='TimeoutOnly', tls="openssl"})
newServer({address="103.2.57.6:853", healthCheckMode='lazy', checkInterval=1, lazyHealthCheckFailedInterval=30, rise=2, maxCheckFailures=3,
lazyHealthCheckThreshold=30, lazyHealthCheckSampleSize=100, lazyHealthCheckMinSampleCount=10, lazyHealthCheckMode='TimeoutOnly', tls="openssl"})

-- DNS cache
pc = newPacketCache(10000)
getPool(""):setCache(pc)
```

どうせなら、ってことで、PublicDNSをIIJ様へ変更し、2つ指定することで冗長性を持たせます。
さらに、GoogleやCloudFlareを追加してマルチベンダー化することも可能です。

まとめ

- dnsdistによるDNSクエリの暗号化は成功
  - DoT、DoH(試してみました)ともにOK
  - 複数サーバ(サービス)を指定した冗長化
  - キャッシュ機能による、同一のレコードの問い合わせ抑止
  - サービス停止時の切り離しと、lazyモードのヘルスチェックによるサービス復活監視
- とはいえ、家庭でサーバを常時起動は非現実的(ラズパイとかでやる？)
- ホームルータで普通に使える日がくることを期待