フルリゾルバーにおける DDoS対策機能の評価

阿波連 良尚（JPRS）
小松 吉基（HOTnet）
植松 利亮（HOTnet）

本セッションの内容

- フルリゾルバーに実装されているDDoS対策機能について、国内ISP 9社と共同で検証実験を行った
- JPRSの阿波連からは、実験全体の結果とソフトウェア実装者へのフィードバック結果について説明する
- HOTnetからは、ISPのDNSオペレーターとして実験内容と結果について説明する（スライド12～16）

- 本資料は2021年5月6日にOARC 35（DNS-OARC主催）にて「Evaluation of anti-DDoS features in full-service resolvers」として発表したものに加筆修正したものです

実験用TLD「.jprs」

- JPRSは、実験用TLDとして「.jprs」を運用している
  - 詳しくは: https://tldlabs.jprs/
- 過去に、国内ISP数社と協力して実験を行った
  - “TLD Anycast DNS Serves to ISPs” (APRICOT 2017)
    - TLDのDNSサーバーをISP内に設置する実験
    - JPRSの野口（当時）が発表
    - https://2017.apricot.net/program/schedule/#/day/9/network-operations-2
  - “Deployment of TLD DNS Anycast node to ISPs for stability and resiliency” (APRICOT 2021)
    - 上記実験の追加的な実験
    - JPRSの松浦と、OPTAGEの矢野氏が発表
    - https://2021.apricot.net/program/schedule-conference/#/day/10/network-operations

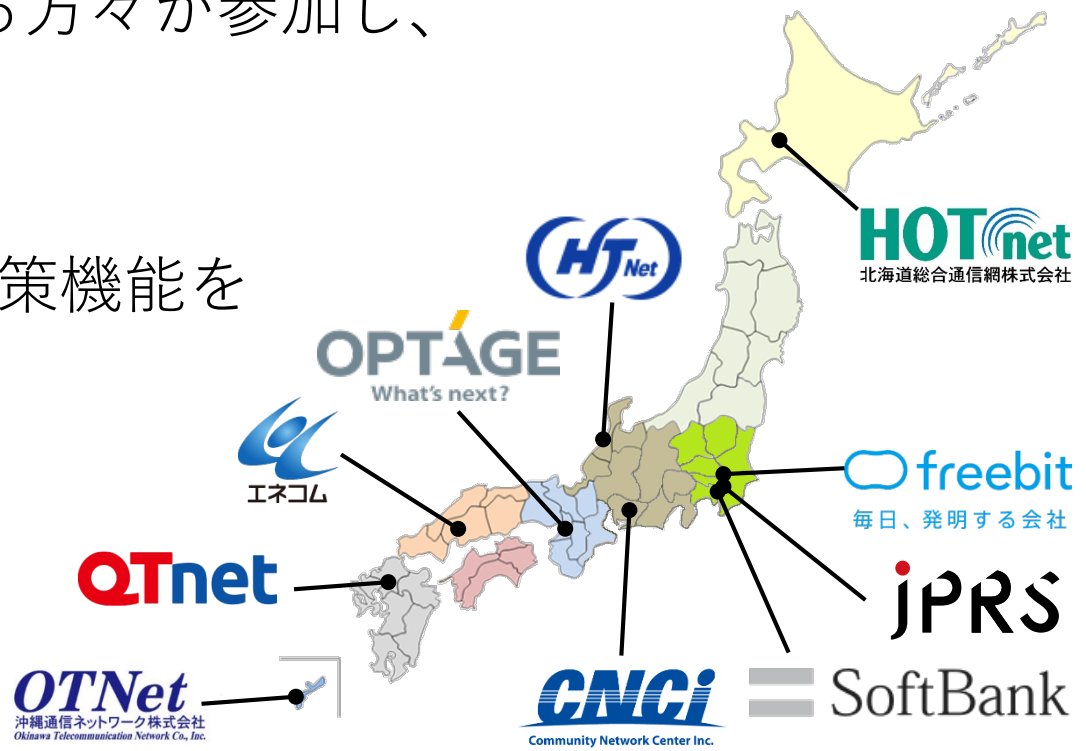
実験の目的

- フルリゾルバーのDDoS対策機能を評価する
  - 実験環境でどのように動くか試してみる
  - 実験して気づいたことや気になったことがあったら実装者（開発元）にフィードバックする
- 試した機能
  - BIND 9（BIND 9.14.5）とUnbound（Unbound 1.9.3）で実験した
  - fetch-limit （BIND 9）・ratelimit（Unbound）
  - serve-stale（BIND 9）・serve-expired（Unbound）
- 試していない機能
  - NSEC aggressive use: .jprsの署名がNSEC3 opt-outなので対象外
  - DNS Cookies: 当時、RFC 7873の改訂がdnsopで進行中

実験計画時点の最新版

実験参加組織

- 国内のISP 9社が実験に参加、共同で実施
  - CNCI・ENECOM・Freebit・HOTnet・HTNet・OPTAGE・OTNet・QTnet・Softbank
  - 各社でDNSサーバーの運用をされている方々が参加し、実験に協力いただいた
- 各社の参加目的
  - サイバー攻撃への対策として、DDoS対策機能を実際に触って検証



事前検証

- マニュアルの記載など机上確認
  - 各機能の概要や、設定パラメーターを洗い出し
- パイロットシナリオを作成して、挙動を確認
  - 実験用のルート・TLD・セカンドレベルドメインの権威DNSサーバーをDNSSEC署名込みで作る手順書を用意し、各社で環境を構築
  - 機能単体で1つずつ、BIND 9・Unboundを設定して実際にクエリを送って動作を確認した
  - 大量のクエリ送信にはdnsperf[1]を利用した
  - この段階ではうまく動いているように見えたが・・・？

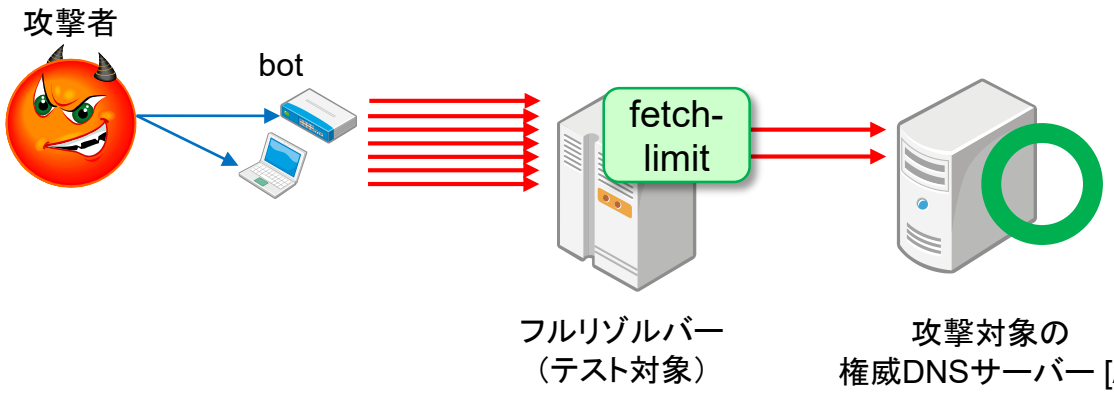
[1] dnsperf: https://www.dns-oarc.net/tools/dnsperf

メインシナリオの作成

- 自社ネットワーク内にbotがいて、ランダムサブドメイン名攻撃を自社フルリゾルバーを通して社外の権威DNSサーバーに行われてしまう想定で作成
- fetch-limit・serve-staleをあらかじめ有効にしてあるとして評価
  - 権威DNSサーバーへの反復検索数を制限できるか
  - 権威DNSサーバーがダウンした場合でもクライアントに応答できるか
- 併せて、攻撃を受けた権威DNSサーバーがより強力な別サービスに引っ越した場合の影響を評価
  - 手元のstaleな（TTLが満了した）キャッシュをいつまで使うか

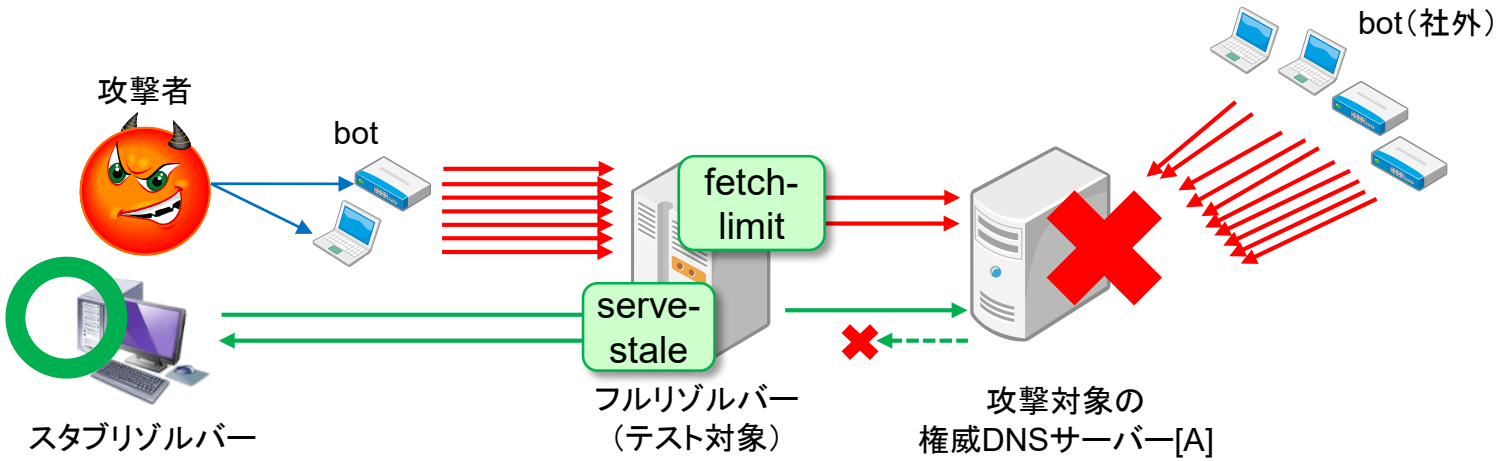
メインシナリオ（1）

- fetch-limitとserve-staleを有効に設定したテスト対象のフルリゾルバーを構築
- rd2020-theme2.jprsというドメイン名が攻撃対象になったと想定
  - インターネット（社外を想定）に攻撃対象の権威DNSサーバーを構築
  - ランダムサブドメイン攻撃を模したクエリを、テスト対象のフルリゾルバーに向けて大量に送信
- fetch-limitが動作し、攻撃対象への同時クエリ数を絞ることを期待



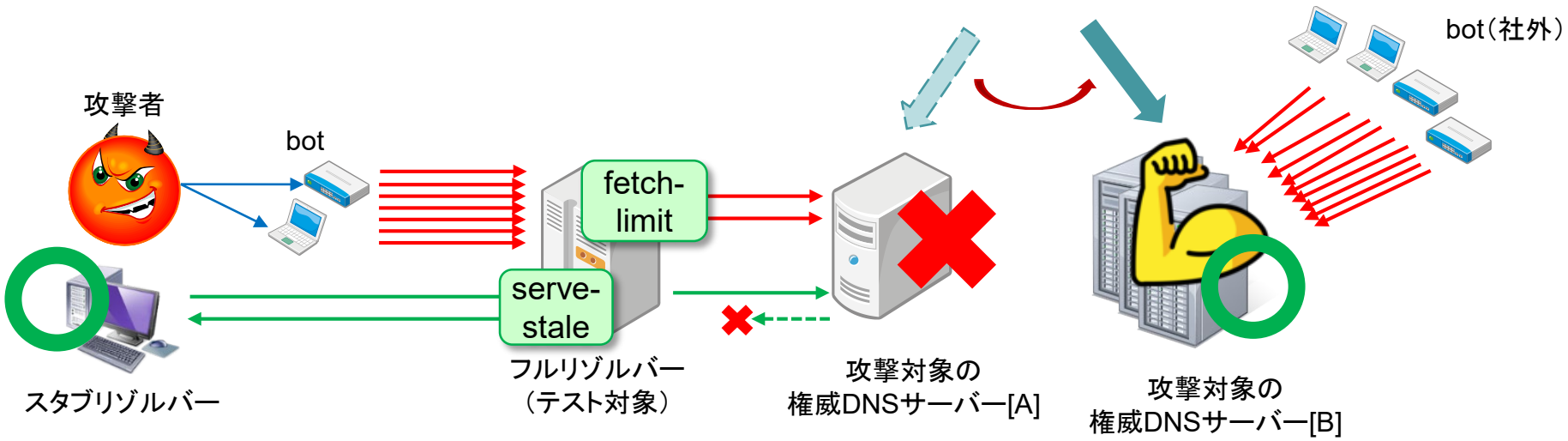
メインシナリオ（2）

- 攻撃対象の権威DNSサーバー[A]が過負荷によりダウン
  - 自社からのクエリは絞り込んでいたが、他からの大量のクエリが影響
  - 実験では、クエリをパケットフィルタでdropすることでシミュレート
- serve-staleにより、TTLが満了した（staleな）キャッシュの情報を使って応答することを期待
  - （攻撃と関係ない）スタブリゾルバーを用意して確認



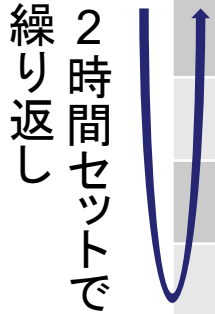
メインシナリオ（3）

- 攻撃対象が、より強力な権威DNSサーバー[B]に乗り換えた
  - DDoSに耐えられる程度にキャパシティがあるのでサービスは復旧
  - 実験では、委任元・先のNS RRsetを切り替えることでシミュレート
- このとき、staleなキャッシュがいつまで使われるか確認
  - ずっと使われるようなら、手動でキャッシュをflushする



権威DNSサーバーの切り替え

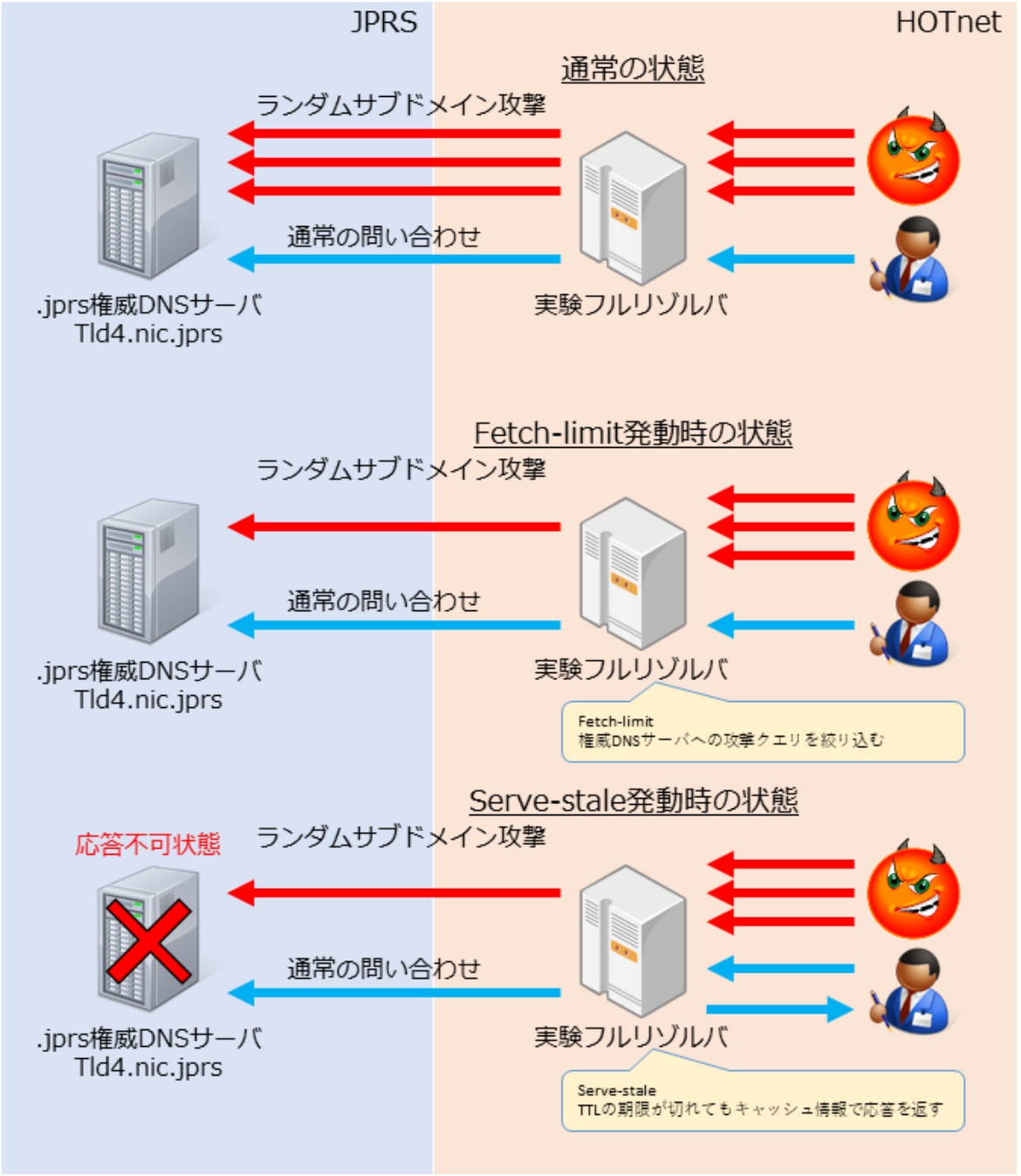
- シナリオの展開で、親ゾーンのNSレコードを切り替えることになる
  - 全組織がスケジュールを合わせて一斉にテストするのは、シフト勤務の関係で調整がつかず・・・
- 定期的にゾーンの内容を差し替えて、自動的にNSレコードを切り替える仕掛けを作成した
  - 併せて、時間帯によって非力なサーバー[A]が応答を返さなくなる仕掛けも用意した
- 各社都合の良い時間に実験できた



時刻	jprsゾーンの内容	サーバー[A]の応答
1:00	サーバー[A]（非力）への委任	応答する
1:30	〃	無応答
2:00	サーバー[B]（強い）への委任	〃
2:30	サーバー[A]（非力）への委任	応答する

実験環境

- 実験用フルリゾルバー
  - OS：CentOS7
  - DNS：bind9.14.5
    - named.conf
      - fetch-limit 設定値：
        - fetches-per-zone 10 fail
      - serve-stale 設定値：
        - stale-answer-enable yes
        - Max-stale-ttl 86400
- 疑似攻撃クライアント
  - OS：CentOS7
  - dnsperf-2.3.2



実験結果　～fetch-limit～

- 疑似攻撃クライアントによるランダムサブドメイン攻撃開始
  - 攻撃トラフィックの発生をフルリゾルバーで確認

- fetch-limitの効果
  →権威DNSサーバに対して攻撃トラフィック量の抑制を確認
  →通常のクエリについては抑制されないことを確認

fetch-limitについて期待した動作結果が得られた

実験結果　～serve-stale～

- 疑似攻撃クライアントによるランダムサブドメイン攻撃継続
  →権威DNSサーバーが無応答な環境を作成

- serve-staleの効果
  →TTLの期限が切れてもキャッシュを破棄せず名前解決が
  できる想定だったが・・・
  →TTLの期限が切れたタイミングから応答が
  status: SERVFAILとなり応答が得られなくなった

fetch-limitとserve-staleは組み合わせて使用できないことが判明

今後の展望　～地域ISPとして～

- 昨今、弊社のお客様より、クラウド基盤を利用していただく上でのセキュリティ要件についてお問い合わせが増加
- 顧客情報を守ることはもちろん、事業設備を守る仕組みについて積極的に検討したい状況
- コロナの影響で出社できる、インシデント対応できる人員が不足しがちな状況から、人の手なしに自動対応できる仕組みが欲しい
- 自動で攻撃を緩和する、攻撃されてもサービスが継続できる機能導入について前向きに考えたい

■ 自社のフルリゾルバに導入したいか？

- ✓ フルリゾルバへのリクエスト数の処理が減ることで攻撃時の負荷が下がる等の効果を期待している
- ✓ 権威DNSへの負荷の軽減に貢献できる点では導入したい
- ✓ エンドユーザへの影響・副作用について検証は必須

所感　～実証実験参加～

- 24時間365日の監視・運用部門の2名にて実証実験に参加
  実験準備、スケジュール調整など難航したが、実験参加メンバーの皆様のフォローで実現
  スケジュールなどはかなり融通を利かせていただいた

- 仲間が増えると心強い
  何か技術的につまずいたら、自社だけでなく実験参加メンバーに相談して進められる
  実験参加メンバーにチャットで気軽に相談することが可能
  実際に不明なログが出てチャットで共有したら解決という場面も

- 今後
  積極的に実証実験に参加していきたい

全体の結果: Unbound

- Unboundでは、シナリオの想定通りに動いた
- ratelimitにより、権威DNSサーバーへの並列問い合わせ数が制限されることを確認した
- 権威DNSサーバーから応答を得られない場合でもキャッシュにある情報をもとにTTLが満了していても応答できることを確認した
  - serve-expiredが期待通り動作した

全体の結果: BIND 9

- BIND 9は、想定と異なる挙動だった
- fetch-limitにより、権威DNSサーバーへの並列問い合わせ数が制限されることを確認した
- しかし、権威DNSサーバーから応答を得られない場合にSERVFAILを返していた
  - serve-staleの効果で、手元のキャッシュをもとに応答することを期待していた
  - `rndc dumpdb`で確認すると、TTLが満了しているキャッシュが残っている状態

結果に関するオペレーターからのコメント

- BIND 9でfetch-limitとserve-staleの組み合わせでうまく動かない
  - 前スライドで説明した通り
- Staleなキャッシュが使われ続けることに関する懸念
  - 攻撃を受けて権威DNSサーバーが変わった後、 Staleなキャッシュを使って応答を返し続けないか
  - もし問題があったとしても、個別対応（キャッシュをクリアするなど）することは難しい可能性
  - NS RRsetのTTLに左右されると思われるが、実験シナリオでは挙動を詳しく確かめられなかった

実装者へのフィードバック: NLnet Labs

- Unboundに関しては、機能的に大きな問題は見つからなかった
- `serve-expired-ttl`のデフォルト値が0になっている件を報告
  - この設定項目で、キャッシュがstaleになった（TTLが満了した）後でもどれだけの期間使い続けるかを制御する
  - デフォルト値の0は「期間無制限」: ずっと使い続ける
  - RFC 8767で提案されている1日～3日とは違う
- NLnet Labsからは「RFCの記載に揃えるのは良い考えだと思うが、デフォルト値を変えるとオペレーターに混乱を招く恐れがある」と返答があった
  - NLnet Labsの考えに同意: オペレーター側で設定内容とその効果を理解して使う必要がある

実装者へのフィードバック: ISC

- BIND 9に関しては、機能の組み合わせで問題が見つかった
- `fetches-per-{server,zone}`が`stale-answer-enable`に干渉し、意図したとおりに動かない件を報告
  - staleなキャッシュを使って応答するのではなく、先に権威DNSサーバーへの同時反復検索数の制限に引っ掛かり、その段階でServFail応答が返る
- ISCからは「意図したとおりに動かないのは問題であり、すでにバグとして報告されている」と返答があった
  - ISC GitLab: https://gitlab.isc.org/isc-projects/bind9/-/issues/1712
  - その後、BIND 9.16.13にて修正された（ChangeLog 5573）
  - 引き続き、設定パラメーターの追加など改善が行われている

まとめ

- フルリゾルバーに実装されているDDoS対策機能の評価を行った
- 結果、一部の実装で機能を組み合わせると期待通り動かないことが分かった
- 気づいた点については実装者に連絡を行った

謝辞

- フィードバックを受け付けて適切に対応くださったISCとNLnet Labsに謝意を表します

- 共同実験にご参加・ご協力いただき、実験環境構築から実験結果のまとめまでさまざまなコメントをいただいたCNCI・ENECOM・Freebit・HOTnet・HTNet・OPTAGE・OTNet・QTnet・Softbank各社の皆様に謝意を表します