複数 DNS キャッシュサーバへの並行名前解決による DNS キャッシュポイズニング対策手法

阿久津賢宏[1] 山井成良[1] 金勇[2]

概要：インターネットを運用するうえで DNS(Domain Name System)は重要な部分を担っており，悪意ある攻撃の対象とされる場合がある．攻撃の一種である DNS キャッシュポイズニングは DNS キャッシュサーバに偽の DNS 情報を登録させるというものである．クライアントが偽の DNS 情報を受け取ると意図せず悪意あるサイトに接続してしまい，フィッシング詐欺などの実害が発生する可能性がある．この攻撃への対応策として DNSSEC(Domain Name System Security Extensions)という DNS 応答の検証システムが存在する．しかし DNSSEC は様々な問題を抱えており，DNS 応答の厳密な検証が行われていない場合が多い．

そこで本研究では複数の DNS キャッシュサーバに対し並行に DNS 問い合わせを行い，各 DNS 応答を用いて信頼性の高いリソースレコードを抽出する手法を提案する．DNS キャッシュポイズニングを複数のサーバへ同時に成功させる困難さを利用し，各応答に共通して存在するリソースレコードを信頼できるものとして扱う．こうすることでクライアントは DNSSEC を用いずに信頼性の高い DNS 応答を扱うことができる．また複数 DNS キャッシュサーバへの問い合わせをマルチスレッド化し，並行に行うことで通常の名前解決と変わらない速度を維持することができる．本論文では提案手法のシステムを実装し，様々なドメインの名前解決を行うことで実環境での本システムの可用性を検証する．また本システムを運用するうえでの注意点や問題点をクライアント・DNS サーバの両者の観点から述べ，改善策を考察する．

DNS cache poisoning countermeasure method by parallel domain name resolution to multiple cache servers

TAKAHIRO AKUTSU[1] NARIYOSHI YAMAI[1] YONG JIN[2]

1. はじめに

現在，広く普及するインターネットを正しく運用するうえで DNS (Domain Name System) という技術が重要な役割を担っている．DNS とは，ネットワーク上の通信で必要な数字である IP アドレスと人が扱いやすいドメインを対応付けさせる技術のことを指す．DNS はドメインの委任という仕組みでドメイン名と IP アドレスのマッピングを階層構造に分散化させている．これはドメインの追加などの更新作業を容易にすると共に負荷分散の効果を持つ．こういった工夫により DNS は急速に成長するインターネットに対応し続けている．

しかし，DNS の重要性が増すにつれて DNS が悪意ある攻撃の対象にされる場合も増えており，実際に DNS キャッシュポイズニングといった攻撃が存在する．これは DNS の名前解決を高速化するために用いるキャッシュに偽のドメインと IP アドレスのマッピングを登録させる攻撃である．DNS キャッシュポイズニングが行われるとキャッシュの利用者を悪意あるサービスに誘導することが可能となり，例えばフィッシング詐欺などの実害が発生する可能性がある．このように DNS の正確な名前解決が行われないと目的のコンピュータとの通信が不能となり，目的のサービスを利用できなくなるためインターネットの運用が停止してしまう．

DNS キャッシュポイズニングのような偽の情報を登録させる攻撃に有効な対策として DNSSEC (Domain Name System Security Extensions) という技術が開発された．この DNSSEC は電子署名を用いて DNS 応答の完全性や正当性を検証することを可能する DNS の拡張機能であり，DNS キャッシュポイズニングを完全に防ぐことを可能にする技術として注目された．

DNSSEC を導入することで DNS キャッシュポイズニングを防ぐことが可能になる一方，DNSSEC には問題点がいくつか存在する．DNS は名前解決を行う際，各ドメインの DNS サーバに再帰的に問い合わせを行う．このとき，各ドメインで DNSSEC による検証が行われなければ最終的にユーザーが受け取る DNS 応答の正確な検証を行うことができない．しかし，DNSSEC の検証をクライアントに提供

1 東京農工大学
Tokyo University of Agriculture and Technology

2 東京工業大学
Tokyo Institute of Technology

するためには各ドメインが DNSSEC を採用する必要がある一方，DNSSEC の採用率がいまだ低いという問題点があげられる[1]．これは電子署名の検証に伴う遅延や鍵の付加によるパケットサイズの増加などの DNSSEC の仕様上の問題が存在するためである[1]．

この問題を解決するため，複数の DNS キャッシュサーバに同時に問い合わせを行い，それぞれの DNS 応答を用いて DNS の検証をクライアント側で行うシステムを提案する．このシステムでは問い合わせた複数 DNS キャッシュサーバからの各応答に共通して存在する IP アドレスを信頼できる応答として利用する．このシステムにより，DNS キャッシュポイズニングによる偽の応答をクライアントに利用させる難度を高め，DNS 応答の信頼性の向上と DNS 検証の高速化を目指す．また本システムの実環境での運用を想定し，様々なドメインの名前解決を行いシステムの動作結果を検証しつつ，システムの可用性の向上について検討および議論を行う．

2. 背景

2.1 ネットワークにおける DNS の重要性

ネットワークに接続されるコンピュータは互いに識別するために IP アドレスという固有の数字列を設定する．現在インターネットを運用する上で用いられているプロトコルの TCP/IP では，この IP アドレスを用いてパケットの通信を行っている．IP アドレスは例えば「198.51.100.0」といった数字列で表現されるため人間にとっては非常に覚えにくく扱いづらい．このためユーザーがネットワークを利用する際に通信相手を直接指定しなければならない場合，IP アドレスを直接利用するのは難しい．そこで TCP/IP ではネットワークに接続したコンピュータにそれぞれホスト名という文字列の識別子を割り当て，ホスト名を IP アドレスに変換することで人間に扱いやすくしている．

DNS ではホストを管理する組織が存在し，組織下に存在するコンピュータのホスト名と IP アドレスの対応を示すデータベースを管理する．この組織を階層的に配置し，組織毎に固有であるドメイン名から対象の組織をたどることで効率的に目的の IP アドレスを得ることができる．またホスト名と IP アドレスの対応を各組織で管理すればよく，組織を増やすことで負荷の分散を可能にしている．また管理される組織が異なれば同じホスト名が存在することも可能であり，名前の自由度も上昇させることができる．

現在私たちがネットワークに接続しアプリケーションを利用できているのは DNS の技術が存在するためである．また，DNS はネットワークに接続するコンピュータの IP アドレスを検索するだけでなく，メールアドレスの名前解決やホスト名の逆引きなどの役割も果たしており，ネットワークを運用するうえで重要な役割を果たしている．

2.2 DNS の仕組み

2.2.1 DNS の動作

ここから DNS の具体的な動きを確認するために DNS サーバについて述べる．DNS サーバは権威サーバと DNS キャッシュサーバの 2 種類存在する．権威サーバとはあるゾーンの管理する名前空間のドメインと IP アドレスの対応情報であるリソースレコードが登録されるサーバのことを指す．権威サーバは自身の子ゾーンの権威サーバの IP アドレスをリソースレコードとして保持しているため，クライアントのコンピュータ上に存在する DNS リゾルバはこの権威サーバをたどることで目的のリソースレコードを得ている．

一方，DNS キャッシュサーバとはクライアントと権威サーバの中間に位置するサーバであり，クライアントから送られる DNS の問い合わせを仲介する役割を持つ．DNS キャッシュサーバは権威サーバから受け取ったリソースレコードを一時的にキャッシュとして保持する．この仕組みは複数のクライアントが同一の DNS キャッシュサーバを利用することで，DNS の問い合わせが重複した際素早くリソースレコードをクライアントに返すことでできる．DNS キャッシュサーバはクライアントのリゾルバに代わり実際に権威サーバへの問い合わせを行うためフルサービスリゾルバとも呼ばれ，クライアントのコンピュータ上に存在するリゾルバをスタブリゾルバと呼ぶ．

ここまで DNS サーバとリゾルバについて述べた．次に，クライアントから DNS 要求が発生してから DNS 応答としてリソースレコードが返され名前解決が終了するまでの DNS の動作を確認する．ここでは説明を簡単にするため，クライアントがドメイン”www.example.jp.”の名前解決要求を行った場合について考える．またルートドメインは”jp”ドメインを委任しており，”jp”ドメインは”example.jp”ドメインの委任を行っているものとする．またホスト”www”は”example.jp”ゾーンに属するものとする．DNS 名前解決の流れについて以下の図 2-1 と共に示す．

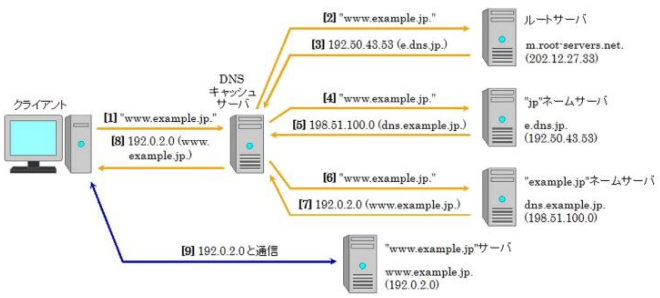

図 2-1 “www.example.jp.”の名前解決の流れ

このような流れでクライアントのスタブリゾルバは目的のリソースレコードを得ることができる．DNS キャッシュサーバに対象のリソースレコードが保存されている間は

同様の DNS クエリに対して権威サーバへの問い合わせを行うことなく応答を返すことができ，DNS の高速化とネットワークの負荷軽減を行っている．また名前空間の委任を行うことで，権威サーバは全てのドメインやホスト名を管理する必要がない．つまり DNS 情報の更新に関して親ゾーンの権威サーバへの更新が必要なく，負荷の集中を防ぎつつネットワーク管理を効率化している．

リソースレコードの更新は DNS キャッシュサーバにも通知されることが無いため，キャッシュされたリソースレコードの整合性を保つためにキャッシュされたリソースレコードには有効期限が設定されている．有効期限が過ぎたリソースレコードは削除され，対象のドメインの名前解決要求が発生した場合は再度権威サーバへの問い合わせを行う．

2.3 DNS キャッシュポイズニング

2.1 節で述べたように DNS の可用性はネットワークを維持するうえで必要不可欠となった．ネットワークの拡大とともに DNS が攻撃対象となることが増えている．DNS への攻撃の一つに DNS キャッシュポイズニングというものが存在する．DNS キャッシュポイズニングとは DNS キャッシュサーバに偽のリソースレコードを登録させる攻撃である．DNS キャッシュポイズニングされた DNS キャッシュサーバにクライアントが問い合わせを行うと登録された偽のリソースレコードがクライアントに送信される．その結果クライアントは偽のリソースレコードに書かれた IP アドレスとの通信を開始してしまう．つまりクライアントは意図せず悪意あるサービスへ誘導されてしまう．以下では DNS キャッシュポイズニングを行う手法の中から従来の DNS キャッシュポイズニング手法とカミンスキー型攻撃手法について述べる．

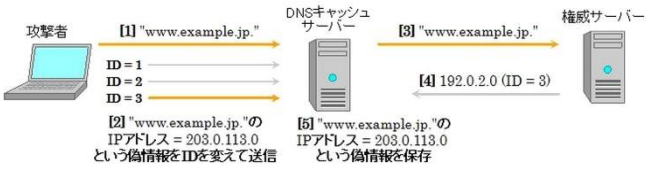

図 2-2 従来の DNS キャッシュポイズニング手法の流れ

この攻撃手法では DNS キャッシュサーバが偽装対象であるドメインのリソースレコードを保持している場合，偽装 DNS 応答を登録させることができない．この性質を利用して DNS キャッシュサーバが保持するリソースレコードの有効期限である TTL(Time To Live)を長く設定することで攻撃の間隔を長くさせ，攻撃の成功率を下げることができると考えられていた．

2.3.1 カミンスキー型攻撃

2.3.1 項で述べた通り従来は TTL を長くすることで DNS キャッシュポイズニング成功率を下げることができると考えられていた．しかし 2008 年 7 月に Dan Kaminsky 氏によって新たな効率の良い DNS キャッシュポイズニング手法が提案された[2][3]．この手法は偽装対象のドメインに関するリソースレコードが DNS キャッシュサーバに保存されていたとしても繰り返し攻撃を行うことができる．つまり従来の対策が有効でないと判明し，新たな対策手法が考えられるきっかけとなった．この攻撃手法のことをカミンスキー型攻撃と呼ぶ．以下の図 2-5 にカミンスキー型攻撃の流れを示す．またここでは攻撃者が偽装するドメイン名を”www.example.jp.”, 偽装する IP アドレスを”192.2.2.20”とする．

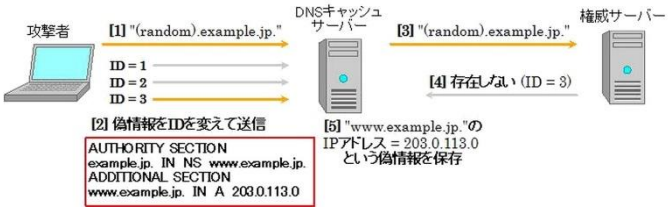

図 2-3 カミンスキー型攻撃の流れ

この攻撃は偽装したいリソースレコードを DNS 応答の付加情報としているため DNS キャッシュサーバに攻撃対象のドメインに関するリソースレコードが存在しても行うことができる．つまり図 2-3 [1]で問い合わせるドメインのホスト名を変えることでキャッシュの TTL に関係なく繰り返し行うことができる．したがってカミンスキー型攻撃は従来の DNS キャッシュポイズニング手法と比較して攻撃の成功率が高くネットワークの脅威となる攻撃であるといえる．

2.3.2 DNS キャッシュポイズニングへの対策

カミンスキー型攻撃の発見から DNS キャッシュポイズニングの早急な対応が必要となった．以下にカミンスキー型攻撃を含む DNS キャッシュポイズニングに有効な対策を示す．

ソースポートランダマイゼーション

これは UDP ヘッダの送信元ポート番号を DNS クエリで固定するのではなく，通信ごとにランダムな値に変えることで偽装パケットがキャッシュされる確率を下げる手法のことを指す．一般に DNS キャッシュポイズニングを成功させるには DNS ヘッダ ID を予想する必要がある．これは 2.4.1 項で述べたように「送信元 IP アドレス」,「UDP ヘッダのポート番号」が問い合わせパケットと応答パケットで一致している場合に成り立つ．したがって DNS ヘッダの ID 以外にこれらの内容を予想できないものにすることができれば偽装パケットが正規の物であると誤認される確率が減少する．

DNS ヘッダ ID は 16bit の数であるため 65536 通りの値を持つ．また送信元ポート番号も同様に長さ 16bit の数で

あり 65536 通りの値を持つ．つまりソースポートランダマイゼーションを行い DNS ヘッダ ID と送信元ポート番号共にランダムな値を用いると攻撃者は 4,294,967,296通りから正しい数の組み合わせを予想する必要がある．ソースポートランダマイゼーションはポート番号を固定する場合に対して DNS キャッシュポイズニングを成功させる確率を低下させることができるとわかる．

DNSSEC

DNSSEC は電子署名を用いてリソースレコードの正当性を検証する DNS の拡張機能であり，DNS キャッシュポイズニングの根本的な解決策として注目された．

2.4 DNSSEC の動作

DNSSEC とは電子署名を用いてリソースレコードが改ざんされていないことを検証する仕組みである[4]．以下に DNSSEC を用いた場合の DNS 通信と検証の流れを示す．

このように各ゾーンはリソースレコード検証用の鍵と子ゾーンの鍵検証用の鍵の 2 種類の鍵を用意し，親ゾーンから順に鍵とリソースレコードの検証を繰り返すことでリソースレコードの正当性を確保している．すべての権威サーバが信頼できるサーバであるとは限らないため，DNSSEC では子ゾーンの鍵の署名を親ゾーンの権威サーバが行うことで，ある信頼できる権威サーバから順に子ゾーンの信頼性を検証できるようにしている．このような親ゾーンが子ゾーンの信頼性を担保する仕組みのことを信頼の連鎖と呼ぶ. また信頼できる権威サーバであると設定され，信頼の連鎖の起点となる権威サーバのことをトラストアンカーと呼ぶ．

2.4.1 DNSSEC の問題点

DNSSEC には様々な問題が存在し[1], 以下でこれらの問題について述べる．

DNSSEC の普及率

DNSSEC は信頼の連鎖がトラストアンカーからすべてつながることで正確な検証を行うことができる．つまり権威サーバへの問い合わせを繰り返す名前解決においてすべての権威サーバが DNSSEC に対応し正常に検証を行っている必要がある．現在登録された 1514 個の TLD の内ルートサーバに署名を登録しているものは 1386 個あり，TLD 全体の約 91.5%が署名を親ゾーンに登録している[5]．一方，人気 Web サイトのドメイン 100 万[6]の DNSSEC の普及率調査[7]によると署名済みのドメインは全体の 1.46%であった．この結果から TLD の権威サーバによる DNSSEC 検証は行われている一方，信頼の連鎖が名前解決終了まで続いていない可能性が高いことがわかる．

パケットサイズの増加

一般に DNS は 512Byte までの応答サイズをサポートしている．しかし DNSSEC の検証に用いられる鍵は 126Byte 以上あるものが多く，DNSSEC を採用すると DNS 応答が 512Byte を超える可能性がある．実際 DNSSEC に対応した DNS 応答は 1500Byte を超えることが多い[1]．UDP を用いて DNS 通信を行う際に応答サイズが 512Byte を超える場合 EDNS0 (Extension Mechanisms for DNS) という DNS の拡張機能を導入する必要がある. EDNS0 に対応していない場合 TCP 通信に切り替える必要があるが, TCP では 3 ウェイハンドシェイクといったオーバヘッドが発生しサーバへの負荷や通信の遅延が発生してしまう可能性がある．

相互運用性問題

1 フレームで送れるデータの最大値である PathMTU より大きいパケットは経路上でフラグメント化を起こす．DNSSEC を導入し DNS 応答サイズが増加すると DNS 応答がフラグメント化を起こす場合がある．しかしフラグメント化した DNS 応答はセキュリティのためファイアウォールによってブロックされてしまう場合がある. 実際に 24 種類のファイアウォールのうち 512Byte 以上の DNS 応答に対応したものは 4 つのみであった[1].

中間者攻撃問題

DNSSEC の導入により DNS 応答のフラグメント化が発生すると，構造上ポート番号や DNS ヘッダ ID などのヘッダ情報が最初のフラグメントに含まれる．これを悪用し 2 つ目以降のフラグメントを偽装する第 2 フラグメント便乗攻撃により DNS キャッシュポイズニングを行うことができる[8][9]．つまり DNSSEC の導入により新たな攻撃が可能となる場合がある．

リフレクション攻撃

リフレクション攻撃とは攻撃対象の IP アドレスを用いて DNS 要求を送ることで攻撃対象に大量の応答を効率的に送り込むことができる DoS 攻撃である. DNSSEC の導入による DNS 応答サイズの増加は効率的なリフレクション攻撃を可能にしてしまう. つまり DNSSEC の普及によりリフレクション攻撃が増加する恐れがある．

DNSSEC の完全な普及は DNS キャッシュポイズニングに対する根本的な解決法である一方，上記のような問題点も多く存在する．そこで本論文では DNSSEC を利用せず DNS 応答の検証を行うシステムを提案する．

3. 複数 DNS キャッシュサーバへの並行名前解決

3.1 概要

DNS キャッシュポイズニングの脅威が拡大する一方，DNS キャッシュポイズニングの根本的な解決法である DNSSEC には様々な問題が存在する．これらの問題の多くは DNSSEC 導入によるパケットサイズの増加や負荷の増大が原因となっている．しかしパケットサイズの増加や負荷の増大は DNSSEC の仕様により発生しているため DNSSEC を導入した状態での解決が難しい．

そこでこの問題を解決するために，複数の DNS キャッ

シュサーバへ並行に DNS 問い合わせを送信し各応答から信頼性の高いリソースレコードを抽出する手法を提案する．一般に名前解決を行う際にクライアントは対応する IP アドレスを一つ得ることができれば対象ドメインのサーバと通信を行うことができる．DNS の応答は DNS ラウンドロビンにより複数の A レコードを含む場合がある．そこでこの提案手法ではクライアントが同時に異なる DNS キャッシュサーバへ同一の問い合わせを行い，全応答に共通して存在する A レコードを信頼できるリソースレコードとして抽出する．また全応答に共通して存在する A レコードが存在しない場合はポップアップ形式でクライアントに警告を行う．この提案手法に対して偽装したリソースレコードを利用させる場合，攻撃者は複数の DNS キャッシュサーバへ同時に DNS キャッシュポイズニングを行う必要がある．つまり提案手法を用いることで偽装したリソースレコードを通信に利用してしまう確率の低下を期待できる．また提案手法では DNSSEC を用いないため遅延や負荷を抑えつつ DNS 応答の検証を行うことができる．

今回の研究では提案手法に対して DNS キャッシュポイズニングを行う困難さの検証を行うことができなかった．本論文では提案手法の実装を行うとともに，提案手法を用いて人気サイトのドメインの名前解決を行い，すべての応答が偽装されていないものと仮定した場合の可用性について示す．

3.2 システムの動作設計

本システムでは平行に問い合わせを行う DNS キャッシュサーバを Google Public DNS(8.8.8.8)と Cloudflare のパブリック DNS(1.1.1.1)を用いる．本システムの動作の概要図を以下の図 3-1 に示す．

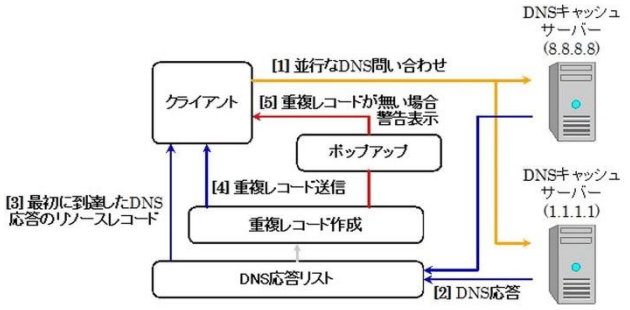

図 3-1 提案手法の動作の概要図

Step.1 同内容の名前解決要求を 8.8.8.8, 1.1.1.1 の 2 サーバへ送信する．この時名前解決要求をマルチスレッド化することで並行に問い合わせを行う．(図 3-1 [1])

Step.2 各 DNS キャッシュサーバからの DNS 応答を DNS 応答リストに追加する．(図 3-1 [2])

Step.3 名前解決におけるクライアントから見た応答速度を維持するため，Step.2 で最初に到達した応答のリソースレコードをクライアントへ送信する．(図 3-1 [3])

Step.4 各 DNS キャッシュサーバからの全応答が到着した後，全応答に共通して含まれるリソースレコードを抽出する．

Step.5 全応答に共通して含まれるリソースレコードが存在する場合，得られたリソースレコードを信頼できるものであるとして再度クライアントへ送信する．(図 3-1 [4])

Step.6 全応答に共通して含まれるリソースレコードが存在しない場合，クライアントが利用しているリソースレコードが危険なリソースレコードであるという内容の警告をポップアップ形式で表示する．

このような流れで信頼できる DNS 応答の抽出を行う．本システムでは 1 つの DNS キャッシュサーバに偽装リソースレコードが登録されている場合必ず警告を表示することができる．一方，DNS ラウンドロビンや権威サーバの設定によって正規のリソースレコードに対しても警告を表示してしまう場合がある．この性質から本システムでは重複するリソースレコードが存在しない場合，リソースレコードをブロックするのではなくクライアントへ警告を表示する．また警告表示にとどめ，リソースレコードを利用するかどうかの判断はクライアントに委ねることで，全 DNS キャッシュサーバへの問い合わせに対して応答が返る前にリソースレコードをクライアントに送ることができる(Step.3)．これによってクライアントから見た名前解決の速度を落とすことなくリソースレコードに信頼性を検証することができる．また複数 DNS キャッシュサーバへ並行に問い合わせた後，一番早く返された応答をクライアントは利用することができるため，単一の DNS キャッシュサーバへ問い合わせた場合の名前解決より早い名前解決が期待できる．

以下の図 3-2 に本システムの流れを表すフローチャートを示す．

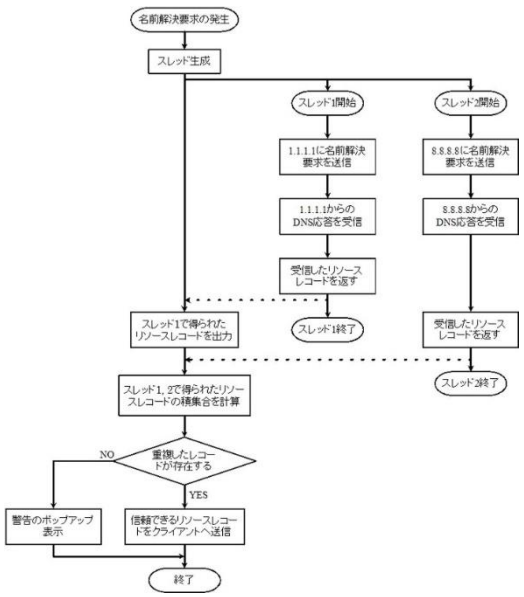

図 3-2 本システムの流れを示したフローチャート

4. 動作確認

4.1 実行環境

本システムの実装および実行を行った計算機の環境をまとめた表を以下の表 4-1 に示す．本システムの実装には Python 3.7 を使用した．

表 4-1 実行環境

OS	Windows 10 Home
CPU	Intel Core i7-7700K 4.20GHz
RAM	16.0GB DDR4
Python IDE	JetBrains PyCharm Community Edition 2019.3

また Python を用いて本システムの実装を行う上で以下のモジュールを使用した．

(1) **dnspython**

DNS 通信やリソースレコードの管理を Python で扱うことができるモジュール．本システムでは動作確認のための名前解決と，システムの可用性確認で用いるドメインリスト作成の際に使用した．

(2) **concurrent.futures**

非同期な並列処理を行うスレッドプールやプロセスプールを Python で扱うことができる Python 標準モジュール．本システムでは複数 DNS キャッシュサーバへの並行な問い合わせを行う際に使用した．また本システムではスレッドプールを用いることで，スレッド数上限の変化や問い合わせを行う DNS キャッシュサーバの数の変更を容易にしている．

(3) **tkinter**

Python で GUI を扱うことができる標準モジュール．本システムではクライアントへの警告としてポップアップ表示を行う際に使用している．

4.2 システムの動作確認

本システムを実装したプログラム”dns_search.py”を用意し，Pycharm 上で実行した際の動作を確認する．問い合わせを行う DNS キャッシュサーバは”8.8.8.8”, “1.1.1.1”の 2 つとする．また名前解決を行うドメインは，”tuat.ac.jp.”と”google.com.”の 2 通りを検証する．

4.2.1 “tuat.ac.jp.”の名前解決

ドメイン”tuat.ac.jp.”の名前解決を行った際の実行結果を以下の図 4-1 に示す．

```
Domain Name : tuat.ac.jp.

first response
165.93.11.4

response from [1.1.1.1]
165.93.11.4

response from [8.8.8.8]
165.93.11.4

reliable response
165.93.11.4
```

図 4-1 ”tuat.ac.jp.”の名前解決を行った際の実行結果

図 4-1 の実行結果では”1.1.1.1”, “8.8.8.8”からの各応答が共通して”165.93.11.4”を含んでいる．つまり”tuat.ac.jp.”に対応する IP アドレスの内”165.93.11.4”が信頼できる IP アドレスであるとして，警告のポップアップをせずに”165.93.11.4”を表示している．

4.2.1 “google.com.”の名前解決

ドメイン”google.com.”の名前解決を行った際の実行結果を以下の図 4-2 に示す．

```
Domain Name : google.com.

first response
172.217.26.46

response from [1.1.1.1]
172.217.26.14

response from [8.8.8.8]
172.217.26.46
```

図 4-2 ”google.com.”の名前解決を行った際の実行結果

図 4-2 の実行結果では”1.1.1.1”, “8.8.8.8”からの各応答内容が異なっており，応答に重複して含まれる IP アドレスが存在しない．最初に返された応答である”172.217.26.46”を表示した後，ポップアップによる警告を表示している．表示されたポップアップ警告を以下の図 4-3 に示す．

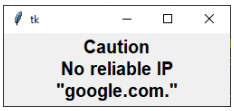

図 4-3 ポップアップされた警告ウィンドウ

以上から DNS 応答に重複するリソースレコードが存在しない場合，正常に警告を表示できていることがわかる．一方，”172.217.26.14”, “172.217.26.46”はともに”google.com.”を示す正常な IP アドレスである．このように DNS キャッシュサーバから正当な応答がある場合も警告が表示されてしまう場合が存在する．本システムでは警告が表示されたドメインのリソースレコードの最終的な利用判断をクライアントに委ねているため，DNS キャッシュポイズニングの有無に関係なく様々なドメインで警告が表示されてしまう

と，危険なリソースレコードと安全なリソースレコードの判断が困難となる．

そこで次節で本システムを用いて有名ドメインの名前解決を行う場合の動作結果を調査する．

4.3 システムの可用性検証

本システムを用いて複数ドメインの名前解決を行い，応答に信頼できる IP アドレスが存在するドメインの割合と，不在応答などのエラーが返される割合を調査する．用いるドメインは“Alexa – Top sites”[6]のランキングトップ 500 ドメインと，“Cisco Popularity List”[10]で提供される 100 万ドメインの計 1000500 ドメインである．ただしこのドメインリストは CNAME による重複が含まれる可能性があるため，まず 1000500 ドメインに対し NS レコードの問い合わせを行い権威サーバの名前を収集し重複なしでリスト化を行う．ここで得られた権威サーバの全ドメインに対しホスト名を”www”に変更した後，システムを用いて名前解決を行う．

権威サーバのドメインはすべてで 337793 個得られた．このドメインに対し名前解決を行った際の動作結果を分類した表を以下の表 4-2 に示す．

表 4-2 337793 ドメインの名前解決に対する動作結果

	エラーなし	エラーあり
信頼できるレコードが存在する	272010 (80.5%)	0 (0.0%)
信頼できるレコードが存在しない	18339 (5.4%)	47444 (14.0%)

表 4-2 から 80%以上のドメインで信頼できる IP アドレスを抽出できている．次に dnspython により出力されたエラー内容を DNS キャッシュサーバ毎に分類した表を以下の表 4-3 に示す．

表 4-3 エラー内容の割合

	“1.1.1.1”	“8.8.8.8”
dns.resolver.NXDOMAIN	42545 (89.7%)	42580 (89.7%)
dns.resolver.NoAnswer	3585 (7.6%)	3586 (7.6%)
dns.exception.Timeout	525 (1.1%)	259 (0.5%)
dns.resolver.NoNameservers	713 (1.5%)	371 (0.8%)

表 4-3 から，エラーが出力されたドメインの 9 割以上が，ドメインが存在しないことを示す NXDOMAIN や応答が存在しないことを示す NoAnswer によるものであるとわかる．これは DNS 応答率向上のため権威サーバ名に対しホスト名を”www”に変更したことにより，存在しないドメインの名前解決要求を送信している場合が原因として考えられる．また存在するドメインの名前空間は常に変化しているため，ドメインリストを取得してから問い合わせを行うまでの間にドメインが消失した可能性も考えられる．

一方，タイムアウトの原因に関してはドメインの存在しないこと以外に通信経路上の問題や，権威サーバをポップする際の遅延などが考えられる．名前解決が遅延しており応答を返す時間が必要となっていることによるタイムアウトの場合，DNS キャッシュサーバへの名前解決要求を再度送信することが問題緩和策として考えられる．

一般に名前解決対象のドメインと対象のサーバが存在し権威サーバが正常な応答を返している場合，DNS 通信においてエラーが表示されることは少ない．つまり表 4-2 においてエラーが発生しておらず信頼できるリソースレコードが存在しない場合，正常な応答に対し警告を表示してしまっていると考えられる．エラーが発生していないという条件下で信頼できるリソースレコードが存在しないドメインは 18339 個存在し，エラーが発生していないドメインの約 6.3%を占める．

5. おわりに

5.1 考察

本研究では DNSSEC の問題点を補う新たな DNS 応答の検証手法を提案した．本システムにより DNS サーバの設定を変更することなく，クライアントの手元で信頼性の高いリソースレコードの抽出を実現している．また問い合わせを行う DNS キャッシュサーバの数を増やすことで攻撃者はより多くの攻撃を同時に成功させる必要があり DNS キャッシュポイズニングの難易度が上がると予想されるため，抽出するリソースレコードの信頼性を高めることができる．

4.3 節で述べた通り，DNS キャッシュサーバが DNS キャッシュポイズニングの被害を受けておらず全応答が正当なものであるという仮定の下，本システムの誤警告率は約 6.3%と低い値を示した．本システムにおいてさらに誤警告を下げることができれば，クライアントに正確な警告を行うことができ可用性が向上するとともに，本システムが DNS キャッシュポイズニングのより有効な対策となる．誤警告率の低下が期待できる対応策を以下に示す．

(1) 問い合わせを行う DNS キャッシュサーバ数の調整

本研究において実装したシステムでは問い合わせを行う DNS キャッシュサーバの数を 2 つに設定した．つまり 2 つの DNS キャッシュサーバに異なるリソースレコードが保存されていると警告を行ってしまう．そこで問い合わせる DNS キャッシュサーバの数を 3 つ以上に設定し 2 つ以上の応答に共通して含まれるリソースレコードを信頼できるものとすることで，すべての DNS キャッシュサーバに

保存されているリソースレコードが異なる場合にのみ警告が行われる．つまり問い合わせる DNS キャッシュサーバの数を増やすことで誤警告率の低下が期待できる．

(2) DNS ラウンドロビンの設定の調整

DNS ラウンドロビンにより 3 つの IP アドレス(a, b, c)が割り当てられたドメインについて考える．このドメインの権威サーバが 3 つのリソースレコード(a), (b), (c)を問い合わせ毎に順に応答している場合，2 つの DNS キャッシュサーバに(a), (b)のような重複する IP アドレスが存在しないリソースレコードが保存されることがある．この状態で本システムの名前解決を行うと警告が発生してしまう．そこで問い合わせ毎に応答するリソースレコードが(a, b), (b, c), (c, a)となるように権威サーバ上で DNS ラウンドロビンの設定を変更すると，2 つの DNS キャッシュサーバにどの応答が返されても必ず重複する IP アドレスが存在するため誤警告がなくなる．こうすることで DNS ラウンドロビンによるサーバへの負荷分散を行いつつ，誤警告率が低下すると考えられる．

5.2 今後の課題

本研究における今後の課題として，複数 DNS キャッシュサーバへの同時 DNS キャッシュポイズニングの困難さの計測による，抽出するリソースレコードの信頼性検証が挙げられる．また，実用化に向けて本システムの警告判定の改善とともに誤警告率の低下も必要である．上記の誤警告率を低下させる対応策の実装・検証とともに，警告に用いるポップアップの改善が必要であると考える．

謝辞

本研究を行うにあたって，適切な指導やサポートを賜った指導教員の山井成良教授と北川直哉助教に感謝いたします．関連研究の情報や技術的な助言をくださった東京工業大学の金勇特任助教に感謝いたします．研究室での日頃のサポートやアドバイスをくださった山井研究室の皆様に感謝いたします．学生としての日常生活を支えて頂いた家族に感謝いたします．

参考文献

[1] Amir Herzberg, Haya Shulman. “Towards Adoption of DNSSEC : Availability and Security Challenges”. IACR Cryptology e Print Archive 2013.

[2] Dan Kaminsky. “It’s The End Of The Cache As We Know It. Black Hat conference”. 入手先 <https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf> (参照 2020-05-16).

[3] “新たなる 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～”. 入手先 <https://jprs.jp/related-info/guide/topics-column/no9.html#a4> (参照 2020-01-20).

[4] “インターネット 10 分講座:DNSSEC”. 入手先 <https://www.nic.ad.jp/ja/newsletter/No43/0800.html> (参照 2020-01-20).

[5] “ICANN Research - TLD DNSSEC Report”. 入手先 <http://stats.research.icann.org/dns/tld_report/index.html> (参照 2020-05-18).

[6] “Alexa – Top sites”. 入手先 <https://www.alexa.com/topsites> (参照 2020-01-20).

[7] 佐藤 一道, 下田 晃弘, 石橋 圭介. “DNSSEC 普及状況アップデート+ランダムサブドメイン攻撃検知手法の紹介”. 入手先 <https://dnsops.jp/event/20150724/dns-summer-day-2015-ksato-1.pdf> (参照 2020-05-18).

[8] A. Herzberg and H. Shulman. "Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org," 2013 IEEE Conference on Communications and Network Security (CNS), National Harbor, MD, 2013, pp. 224-232, DOI: 10.1109/CNS.2013.6682711.

[9] 太田 健也, 鈴木 恒彦. “DNS 第一フラグメント便乗攻撃の追検証と対策の検討”. 第 81 回全国大会講演論文集 2019(1) 443-444, 入手先 <https://ipsj.ixsq.nii.ac.jp/ej/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=196252&item_no=1&page_id=13&block_id=8> (参照 2020-05-17).

[10] “Cisco Popularity List”. 入手先 <http://s3-us-west-1.amazonaws.com/umbrella-static/index.html> (参照 2019-11-18).

[11] 井口 和哉. “DNSSEC における各クライアントでの検証結果の共有による高速化”. 東京農工大学工学部情報工学科 2018 年度卒業論文. 2019.

[12] D. Atkins, JPRS(株式会社日本レジストリサービス)訳. “DNS (Domain Name System)の脅威分析”. RFC 3833 (Informational) Network Working Group, 入手先 <https://jprs.jp/tech/material/rfc/RFC3833-ja.txt> (参照 2020-01-29).

[13] Y. Jin, M. Tomoishi and N. Yamai. "An advanced client based DNSSEC validation and preliminary evaluations toward realization," 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2016, pp. 178-183, DOI: 10.1109/ICTC.2016.7763463.

[14] K. Kakoi, Y. Jin, N. Yamai, N. Kitagawa and M. Tomoishi. “Cache Function Activation on a Client Based DNSSEC Validation and Alert System by Multithreading”. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). Turin, 2017, pp. 37-42. DOI: 10.1109/COMPSAC.2017.78.

[15] Yong JIN, Kunitaka KAKOI, Nariyoshi YAMAI, Naoya KITAGAWA, Masahiko TOMOISHI. “A Client Based DNSSEC Validation System with Adaptive Alert Mechanism Considering Minimal Client Timeout”. IEICE Transactions on Information and Systems. 2017, E100.D 巻, 8 号, p. 1751-1761. DOI: 10.1587/transinf.2016ICP0028.

[16] Y. Jin, M. Tomoishi and N. Yamai. “A Client Based DNSSEC Validation Mechanism with Recursive DNS Server Separation”. 2018 International Conference on Information and Communication Technology Convergence (ICTC). Jeju, 2018, pp. 148-153. DOI: 10.1109/ICTC.2018.8539727.

[17] K. Kakoi, Y. Jin, N. Yamai, N. Kitagawa and M. Tomoishi. “Design and Implementation of a Client Based DNSSEC Validation and Alert System”. 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). Atlanta, GA, 2016, pp. 8-13. DOI: 10.1109/COMPSAC.2016.91.