情報処理学会論文誌 Vol. 50 No. 3 1012–1021 (Mar. 2009)

# セキュリティを考慮した名前解決エージェントの設計と実装

石原知洋[†1] 関谷勇司[†2] 村井純[†1]

Domain Name System（DNS）は，インターネットで広く一般的に利用されている名前解決システムであり，インターネット上におけるアプリケーションの利用に先立ち，ホスト名から IP アドレスへの変換を行っている．しかし近年，DNS に関する攻撃がいくつか報告されている．その代表的なものが DNS spoofing である．DNS の名前解決を詐称することで，正規の IP アドレスとは異なったアドレスを攻撃対象に与え，正規の通信相手に成りすまして通信を誘導することが可能となってしまう．この問題への対策のために DNSSEC が考案されたが，運用コストなどの問題から普及しているとはいいがたい．そこで本研究では，DNSSEC を利用できない環境であっても DNS spoofing に対抗できる DNS 名前解決方式を提案した．また，提案に基づいたプロトタイプ実装を行い，攻撃に対する効果に関して評価を行った．本研究で提案した方式により，DNSSEC が普及していない現状であっても，クライアントに DNS spoofing からの保護を提供することが可能になった．

# Design and Implementation of Security Aware Internet Name Resolution Agent

Tomohiro Ishihara,[†1] Yuji Sekiya[†2] and Jun Murai[†1]

Domain Name System (DNS) is the most frequently and broadly used Name-Resolution system on the Internet. Applications on the Internet use DNS for translating between the hostnames and their IP addresses. In recent years, numbers of DNS attack incidents have been reported. One of the most typical incidents is DNS spoofing. DNS spoofing spoofs the DNS responses, hence attacker could redirect target traffic to the other hosts. DNSSEC is proposed as a countermeasure to this attack. However, because of its operation costs, DNSSEC hasn't been deployed widely. Therefore, we have proposed a new countermeasure against the DNS spoofing, without utilizing DNSSEC. We have also implemented a prototype and evaluated its effectiveness against DNS spoofing. Our proposed system provided protection against DNS spoofing, even if DNSSEC hasn't been widely deployed in today's Internet.

## 1. はじめに

Domain Name System（DNS）は，インターネットで最も利用されている名前解決システムであり，インターネット上のほとんどのサービスは DNS によってホスト名と IP アドレスの相互変換を行うことで通信相手の特定を行い，通信を行っている．このように DNS はインターネット上のサービスを利用するうえで必要不可欠なものであるが，近年，DNS に対する攻撃がいくつか報告されている．

代表的なものが DNS spoofing による攻撃である．DNS spoofing とは DNS の応答を詐称することで，クライアントやサーバに対して正規のものとは異なるデータを与える攻撃である．この攻撃によって，クライアントやサーバは本来通信しようとした相手とは異なる相手に通信を誘導されてしまう（Pharming）．

このような攻撃可能性は以前より指摘されており，対策のため DNSSEC[14] が考案された．DNSSEC は DNS の情報に対して電子署名を施し，DNS のデータが正当なものであることを保証する技術である．DNSSEC は現在，様々なサーバ実装によりサポートされている．

しかし，現在において DNSSEC によって署名を施しているサーバは少ない．ルートネームサーバでは DNSSEC による署名をしておらず，ジェネリックトップレベルドメインでも DNSSEC を採用しているところはまだ存在していない．また，DNSSEC に対応しているクライアントも現時点では普及しているとはいい難い．

ルートネームサーバのように影響範囲が多いドメインや，トップレベルドメインのようにレコード数が多いドメインにおいては DNSSEC を適用することが難しく，事実 DNSSEC のサーバ実装が出回ってからすでに 10 年近く経過するが，いまだにこれら上位ドメインにおいて DNSSEC は普及していない．

そこで本研究では，DNSSEC を利用できない環境であっても DNS の通信を保護可能な DNS 名前解決方式について提案を行う．

本研究で提案した保護方式により，DNSSEC が利用できない環境であっても DNS に対

†1 慶應義塾大学
Keio University
†2 東京大学
The University of Tokyo

する様々な攻撃のリスクを低減することができる．また，クライアント側にシステムを導入するだけで動作させることが可能であり，サーバ側で対応する必要がないため，少ないコストで DNS のセキュリティを確保することが可能となる．

本章では，現在の DNS に対する攻撃について述べ，対処法について DNSSEC などのいくつかの先行研究について説明する．さらに，それらの攻撃に対処するための新しい方式について提案する．

### 1.1 DNS 詐称攻撃

DNS はインターネット上の通信に先立って利用され，ユーザの指定したドメイン名から IP アドレスへの変換を行う．そのため，悪意を持った攻撃者が DNS の情報を詐称することで対象の通信を正規の通信対象と異なる場所に誘導することが可能となる．DNS はデータの到達性を保証しない UDP を基本的に使用し，またサーバおよびクライアント側で相手の状態を持たず，多くの場合 1 回のパケットのやりとりでトランザクションが完結する．そのため，他のインターネット上のプロトコルに比べて攻撃が容易である．

また，DNS は 1 度行った問合せをキャッシュして問合せ回数の低減を行うため，1 度攻撃が成功した場合，キャッシュによりそれ以降の DNS 問合せがすべて詐称されたものになってしまう．正規のレコードのキャッシュ有効期限が短いものであっても，攻撃者は攻撃によってデータだけでなくそのキャッシュ有効期限も改ざんすることが可能である．そのため攻撃者の任意の期間，クライアントに対して詐称したパケットをキャッシュさせそのデータを参照させることができる．

DNS への攻撃は主にサーバからの返答パケットを偽造し，それをクライアントに送信することで行われる．この攻撃は大別すると以下のそれぞれに分類できる．

(1) monkey-in-the-middle 攻撃
　クライアントからの問合せを盗聴し，その問合せに見かけ上適合するように偽造した返答パケットをクライアントに送信する．

(2) Remote Cache Poisoning 攻撃
　外部よりクライアントに対して大量の偽造した返答パケットを送信し，問合せに適合する偽造パケットがクライアントに届くことを期待する．

以下にそれぞれの攻撃について詳細を述べる．

#### 1.1.1 monkey-in-the-middle 攻撃

DNS では，クライアントとサーバはそれぞれの問合せを，トランスポート層の情報であるポート番号と，DNS のパケットヘッダの中に含まれる ID の両方によって識別する．

monkey-in-the-middle 攻撃は，クライアントから出される DNS の問合せパケットを同一ネットワークセグメント上，もしくは通信経路上において盗聴し，その問合せパケットに適合するようにポート番号および ID を偽造した DNS 返答パケットを正式な返答パケットより先にクライアントに送信することで攻撃を行う．経路上で盗聴を行うことに比べて同一セグメント上で盗聴を行う方が容易なため，無線 LAN のリンクや，Ethernet のリンクなどの共有ネットワーク上で行われることが想定される．盗聴を行うことで確実にクライアントに受け入れられるパケットを偽造できるため，攻撃の成功率は高い．

藤原らの実験報告[3),4)] によれば，monkey-in-the-middle を利用した DNS 詐称攻撃は全 DNS トランザクションの 65.3%の確率で成功したという結果が出ている．また実験後の利用者に対する調査で，少なくとも 1 度は DNS 詐称により非正規サイトに誘導されたユーザが 92.3%に上ることも示されている．

以上の結果が示すとおり，monkey-in-the-middle 攻撃はパケットを盗聴するという前提が必要となるが，攻撃をした場合には高い確率で DNS レコードの詐称が可能である．

#### 1.1.2 Remote Cache Poisoning 攻撃

Remote Cache Poisoning 攻撃はクライアントに対して大量の DNS 応答パケットを送信することで DNS 詐称を行う攻撃である．前述の monkey-in-the-middle 攻撃が盗聴によって問合せパケットに適合するポート番号および ID を詐称するのに対し，この攻撃はポート番号および ID を変えつつ大量にパケットを送ることでそのうちの 1 つが問合せパケットに適合した応答パケットとなることを期待する．クライアントが利用するポート番号および ID は 16 bit のため，確率的に見ればパケット 1 つにつき $1/2^{32} = 1/4294967296$ 程度の確率で適合するパケットとなる．

パケット 1 つだけを見れば DNS 詐称の成功率は高くないが，任意のタイミングで攻撃対象に多数の問合せを発行させることができれば，そのうちの 1 つに適合するパケットが生成できる確率は飛躍的に上がる[11)]．これは誕生日攻撃として広く知られている[12)]．任意のタイミングで問合せを発生させる手法としては，外部から DNS の再帰問合せを送信する，HTML の URL 埋め込みを利用し，web サイトや HTML メールを利用して攻撃対象に多数の名前解決を行わせる，などが存在する．

また，いくつかの古い OS および DNS の実装では，ポート番号および ID の選択に偏りがあるため，事前にいくつかの問合せを行わせることで高い確率で問合せに利用するポート番号と ID を割り出すことが可能となる．

以上のように，Remote Cache Poisoning 攻撃は monkey-in-the-middle 攻撃に比べ，個々

の成功率こそ低いが，多数の問合せを行わせることが可能である場合，もしくは問題のある実装が使われている場合には高い確率で詐称をすることが可能である．加えて，monkey-in-the-middle 攻撃は実行するためにクライアントからのパケットを盗聴できる環境にいなければならないことに比べ，Remote Cache Poisoning 攻撃はインターネット上のどの地点からでも攻撃することが可能である．

#### 1.1.3 Kaminsky Attack

Remote Cache Poisoning 攻撃の亜種として，Kaminsky Attack[2] がある．通常 Remote Cache Poisoning 攻撃は 1 つのレコードに関する大量の返答を詐称して送ることで行われる．これに対して Kaminsky Attack は，異なる種類の存在しないレコードに対する問合せを連続で行い，それに対する詐称したパケットを送りつける．通常の攻撃の場合，正規レコードのキャッシュが有効である間は攻撃ができないが，Kaminsky Attack は異なる種類の問合せを発行するため連続で攻撃することが可能となり，結果的に通常より高い確率で DNS の詐称が可能となる．

### 1.2 DNS のセキュリティ拡張

DNS 詐称の問題に対応するために，DNS ではセキュリティ拡張が考案された．DNS は分散データベースという性質上，一元的なセキュリティの確保が困難である．一般的なインターネット上のプロトコルがサーバとクライアント各 1 つずつ，合計 2 つの対象に関して考慮すれば済むことに対して，DNS においては複数のサーバについて考慮しなければならないからである．そのため，DNS のセキュリティ機構には 2 点間のサーバの通信内容を保証する機構だけでなく，リソースレコードの正当性を保証する機構が存在する．前者の通信内容を保証する機構が TSIG[13] であり，後者のリソースレコードそのものを保証する機構が DNSSEC である．

TSIG は，秘密鍵暗号方式を用いて，DNS サーバとクライアントとの間で行われるトランザクションの認証を行う技術である．サーバとクライアントで秘密鍵を共有することで，あらかじめ認証されたクライアント以外からのサーバの利用を制限したり，DNS メッセージの完全性を確認したりできる．

DNSSEC は，公開鍵暗号方式を用いて，リソースレコードの正当性を保証する技術である．まず，あるゾーンに対して，秘密鍵と公開鍵の鍵対を作成する．そして，ゾーンに存在する各リソースレコードに秘密鍵を用いて署名を行い，その署名と公開鍵をリソースレコードとして公開する．名前解決を行うクライアントは，署名と公開鍵を用いて，データの正当性の確認を行う．このようにして，データの正当性を保証する．

この際，配布されている公開鍵の正当性を保証するために，公開鍵自体も，そのデータの起源が保証されている必要がある．すなわち，公開鍵自体が，上位のゾーンの秘密鍵を使って署名されている必要がある．こうして，上位ゾーンが下位ゾーンの公開鍵を署名して，それをゾーンの委譲に沿ってつなげていくことによって，DNS のツリー構造全体のデータを保証する．つまり，上位のゾーンから署名を連鎖させることによって，DNS のツリー構造全体のデータ起源を保証する．

### 1.3 DNS セキュリティ拡張の問題点

DNS のセキュリティ拡張を使うことにより，データの正当性の検証やトランザクションの保護が可能であるが，DNSSEC は前述のように公開鍵自体の正当性を保証するために上位ドメインから署名を連鎖させる必要がある．しかし現時点で，最上位ドメインであるルートゾーンでは署名が行われておらず，その直下のトップレベルドメインでも主だったゾーンでは署名がされていない．これは，DNSSEC の安全な運用のためには一定期間で鍵を変更する必要があり，ユーザ数・影響範囲が大きく，さらに多くの場合で多数のレコードを保持しているこれらのゾーンで運用するのはコストが高いためである．

このため，現状の DNSSEC は限られた範囲でしか運用されておらず，DNS 詐称攻撃などから一般の DNS 通信を保護することはできない．

### 1.4 DNS cookie

DNS 詐称への対策として提案されているもう 1 つの方式に DNS cookie[5] がある．DNS cookies は，DNS の問合せおよび応答に HTTP で使われるような cookie を追加し，各トランザクションでその cookie を確認することによって DNS 詐称の可能性を減らす方式である．

しかし，この方式はプロトコルの拡張であり，クライアントおよびサーバの両方において対策を施さなければならないため，DNSSEC と同じく普及させるのが難しく，また，プロトコルそのものもまだ議論中であり定まっていない．

そこで本研究では DNS 詐称攻撃の特徴に着目し，DNSSEC が利用できない環境においても DNS 詐称攻撃の成功率を大きく下げることが可能な DNS 名前解決エージェントの提案を行い，そのプロトタイプの設計・実装および攻撃に対する評価を行った．

## 2. 問題解決へのアプローチ

上記に述べた DNS への攻撃に対処するためには，以下の 3 つの事柄が必要となる．

(1)　攻撃を検知する

現在の DNS 実装では，クライアントは問合せに対応する単一の応答パケットが来た

時点でその返答を正しいものと見なして扱う．そのため，攻撃パケットが正規パケットより先に到達した場合には攻撃パケットのデータを利用し，後から届いた正規のパケットはいっさいの処理をせずに捨ててしまう．

そこで本研究では，1 つの応答パケットのみを受信するのではなく，一定期間パケットを受信し続け，問合せに対応するパケットが複数送られているかどうかを検出する．2 個以上の応答パケットが届き，かつそれぞれのパケットが異なる場合に何らかの攻撃がされたと判断する．

正規のサーバが何らかの理由で応答できない状況で monkey-in-the-middle 攻撃によって攻撃された場合，クライアントには偽装したパケットしか届かない場合も考えられる．この場合においても，DNS ID が異なる応答パケットが届いた場合には攻撃を検知できる．さらに DNS ID も合致する場合も考えられる．この場合，本研究が提案する方式ではカバーできないが，これは稀有な例であると考えて本研究の範囲外とした．

(2)　正しい応答を検出する

上記の検出により複数の異なる返答パケットが寄せられていることが判明した場合，それらパケットの中から正しい返答パケットを抽出する必要がある．

本研究では，以前の問合せ履歴の参照や，ブラックリストの参照，同一問合せの複数回発行などで正しい応答パケットの判断を行った．また，それらを補助する形でユーザの操作による問合せ結果の選別も行えるようにした．

(3)　攻撃があったことをユーザに通知する

monkey-in-the-middle 攻撃などは，以前の章で述べたとおり，攻撃者のネットワーク環境に前提が必要であるが，攻撃の成功率はきわめて高い．そのため，正しいパケットを選別するだけでなく，攻撃があったことをユーザに対して通知をし，なんらかの対策（攻撃者のローカルネットワーク上からの排除など）を促すことが重要である．

そこで本研究で提案する方式は，ユーザクライアント上で動作するエージェントとして実現する．このエージェントはクライアント上で動作するすべてのアプリケーションから発行される DNS 問合せを処理し，攻撃が行われた際はユーザに通知する．

また，これらの機能に加え，既存のアプリケーションに対して変更を加えることなく動作することが可能であるという互換性も必要である．本研究では，以上に述べた機能を満たすシステムを設計した．

## 3. 設　　計

前章までの議論をふまえ，本システムの設計ならびに実装を行った．前述したとおり，既存のアプリケーションがそのまま利用でき，かつユーザに対して通知を行う必要があるため，本システムは名前解決エージェントとしてユーザクライアント上で常駐動作し，既存の DNS プロトコルによる問合せを受け付けて名前解決処理を行う．

また，各クライアント上で名前解決を行うことにより，従来では DNS キャッシュサーバが攻撃された場合，影響範囲がその DNS キャッシュサーバを利用するクライアント全域にわたっていたものを，本研究では影響範囲をそれぞれ単独にすることができる．DNS キャッシュサーバを共有しないことで，キャッシュによる効果が減ることも考えられるが，Jung らが行った DNS キャッシュ効果の分析に関する研究[17] によれば，多数のクライアントがキャッシュを共有することによる効果は 10%〜20%にすぎず，全体的な問合せ量からみれば大差がないことが示されている．

### 3.1 攻撃検知

本システムでは，問合せに対応する応答を受けた後も，しばらくの間他の応答を待機することによって攻撃を検出する．ここで，その待機時間をどの程度の長さにするかが問題となる．長時間受信し続けることで攻撃を検出できる確率は増加するが，その分名前解決に必要な時間は増加してしまう．

そこで，適切な待機時間の算出のため，慶應大学湘南藤沢キャンパス内に設置された名前解決を行う DNS キャッシュサーバ上と，東京大学内で多くのユーザが利用する DNS キャッシュサーバ上の 2 カ所において DNS 問合せ応答時間の統計をとった（**図 1**，**図 2**）．

ユーザが多い昼間と少ない夜間では DNS 問合せの絶対数が異なる．したがって，統計は偏りをなくすため，ある 1 日に行われたすべての問合せのトランザクションの中から 10,000 個を無作為抽出して行った．

統計結果より，慶應大学湘南藤沢キャンパスで 98%，東京大学で 99%を超える応答パケットが DNS キャッシュサーバが問合せパケットを送信してから 1 秒以内に返答されていることが分かった．

また，ユーザが許容可能な待機時間に関しては，いくつかの先行研究がなされている．FIONA2004 [15] によれば，ユーザにとって許容できる web の応答時間は 5 秒から 15 秒程度としている．Hoxmeier2000 [16] では，ユーザの web に対する満足度は 9 秒を過ぎたあたりから低下するとしている．

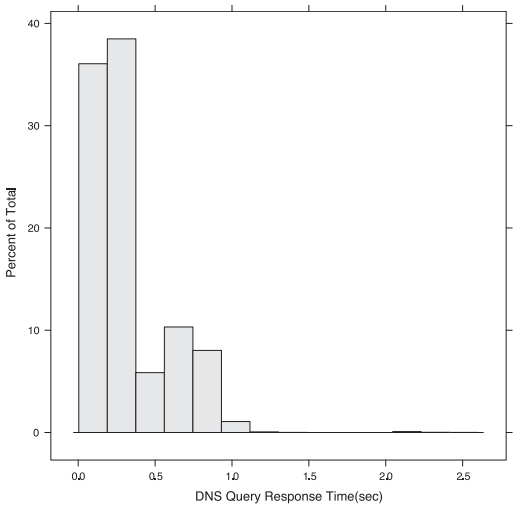



**図 1**　慶應大学湘南藤沢キャンパス設置のネームサーバにおける外部問合せ応答時間の分布（百分率表示）
Fig. 1　Histogram of DNS queries RTT on nameserver in KEIO Shonan Fujisawa Campus (Percentail).

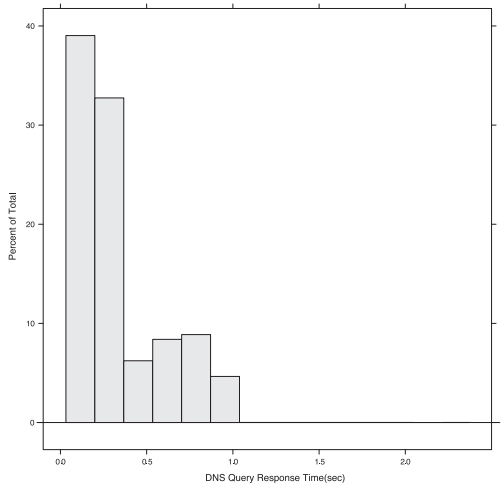



**図 2**　東京大学設置のネームサーバにおける外部問合せ応答時間の分布（百分率表示）
Fig. 2　Histogram of DNS queries RTT on nameserver in the University of Tokyo (Percentail).

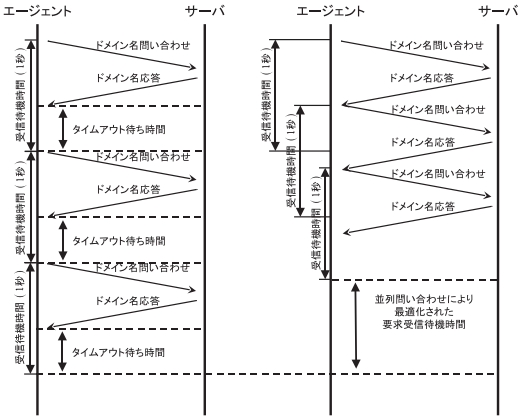



**図 3**　並行した問合せによる待機時間の低減
Fig. 3　Reducing RTT using parallel query.

DNS の応答時間は一般的に 20 msec から 120 msec 程度といわれ，図 2 からも大体が 300 msec 以下ということが読み取れる．

以上から，本システムでは各問合せごとに 1 秒の受信待機時間をとることにした．本システムを導入することで応答時間は数百 msec ほど増加するが，これはキャッシュが存在しない最初の問合せの場合のみである．

また，DNS は再帰的に DNS ツリーをたどりつつデータを検索するため，1 つの名前解決における各問合せごとに 1 秒ずつ待機した場合，全体で数秒ほど待たなくてはならない．そこで本研究では，受信待機と並行して，問合せに対する回答が帰ってきたタイミングでその結果をもとにした問合せを行い，全体の問合せ時間を低減させる（**図 3**）．受信待機中に攻撃を検出した場合，そのドメイン以下の問合せ結果をすべて破棄し，正規パケットと判断された回答結果をもとに問合せを再開する．

本システムでは UDP の問合せのみを対象とする．TCP を用いた DNS 問合せの詐称は TCP 自体のセッションを詐称することで行われる．TCP の詐称および対策に関しては OS のトランスポート層での議論になると考え，本研究の対象外とした．

### 3.2　正規パケット選択

問合せに対応する応答パケットを複数受信した場合，その中から正しいパケットを選択する必要がある．本研究では DNSSEC 署名の有無，過去の検索履歴の参照，再度の問合せな

どで正しいパケットを判別する．

正規パケット判別のために，本システムは問合せの履歴を一定期間保持する．問合せの履歴は問合せ結果のデータだけではなく，その問合せを行ったときに複数の回答があったかどうか，すなわち攻撃されたかどうかも記録し，判断の基準とする．

以下のルールに従ってパケットの正当性を判断する．

(1) 応答パケットのうちの 1 つが DNSSEC で署名されており，その署名の正当性が検証された場合，その応答パケットを正当なものであると判断する．

(2) いずれの応答パケットも署名されていなかった場合，過去の問合せ履歴を参照し，過去に攻撃されたことのない問合せ結果と同じ内容を持つ応答パケットを正しいものとして扱う．

(3) 履歴が存在しなかった場合，もしくは攻撃された履歴しかなかった場合，その問合せを再度行う．問合せした結果応答パケットが 1 つしかなく，その 1 つが前回の問合せに含まれている場合は，そのパケットが正しいものであると判断する．

(4) 再度の問合せでも複数の回答が帰ってきた場合，monkey-in-the-middle によって攻撃されていると判断し，クライアントに SERVFAIL エラーを返答する．ユーザに対して monkey-in-the-middle 攻撃がされていることを通知し，管理者などに相談し対処を行うことを促す．

また，上記のいずれの場合でも，攻撃によって複数の応答を検出した場合，ユーザに対してその旨を通知して注意を促す．

## 4. 評　　価

本研究で提案した名前解決エージェントが攻撃に対して適切な効果があるかどうか実際にエージェントに対して攻撃を行い評価を行った．本章では，その評価結果について述べる．

### 4.1 評 価 実 装

本方式の評価のため，ユーザクライアント上で動作するプロトタイプを実装した．実装は Windows XP のクライアント上で動作するエージェントとして行った．本エージェントはユーザクライアントに常駐し（**図 4**），クライアントが利用する DNS サーバの設定を本エージェントにすることによって動作する．

### 4.2 Remote Cache Poisoning 攻撃に対する評価

Remote Cache Poisoning に関しては，1 つの攻撃対象に関する成功率が低いため，実験で有意な値を得ることが難しい．そこで，Remote Cache Poisoing 攻撃についてモデル化

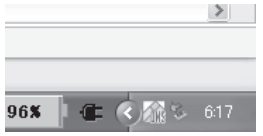



**図 4**　クライアントに常駐するエージェント
Fig. 4　The agent which resides on client.

をし，そのモデルに従って評価を行った．

Remote Cache Poisoning 攻撃が成功するためには，ある問合せを発行してから正規の応答パケットが来る前に，問合せパケットのポート番号と ID に適合した偽造パケットが届く必要がある．

使用するポートの数を $Q$，1 つの攻撃対象に 1 秒間に送るパケットの数を $R$，攻撃を送る対象の数を $N$，問合せを発行してから正規の問合せが戻るまでの時間を $T$ とすると，DNS の ID は 16 bit なので，複数の攻撃対象のうち 1 つ以上に攻撃が成功する確率 $P_s$ は

$$P_s = 1 - \left(1 - \frac{RT}{2^{16}Q}\right)^N \tag{1}$$

となる．連続して攻撃を行うことを考えた場合，対象はキャッシュ有効期限が切れるまでは同じ問合せをしないため，攻撃が行える回数は攻撃時間に比例し，キャッシュ有効期限に反比例する．そのため，連続して攻撃した場合の確率 $P_{sv}$ は，攻撃期間を $V$，キャッシュ有効期間を $A$ とすると

$$\begin{aligned} P_{sv} &= 1 - (1 - P_s)^{\frac{V}{A}} \\ &= 1 - \left(1 - \left(1 - \left(1 - \frac{RT}{2^{16}Q}\right)^N\right)\right)^{\frac{V}{A}} \\ &= 1 - \left(1 - \frac{RT}{2^{16}Q}\right)^{\frac{NV}{A}} \end{aligned} \tag{2}$$

となる．本研究で提案した方式を利用した場合，Remote Cache Poisoning 攻撃が 2 回連続で適合しなければ成功しないため，その確率 $P'_{sv}$ は以下のようになる．

$$P'_{sv} = 1 - \left(1 - \left(\frac{RT}{2^{16}Q}\right)^2\right)^{\frac{NV}{A}} \tag{3}$$

以上の結果をもとに，ある特定の状況を仮定し，従来の名前解決方式と本研究で提案した

**表 1**　Remote Cache Poisoning 攻撃の評価に使用したパラメータ
Table 1　Evaluation parameter for Remote Cache Poisoning Attack.

| | |
|---|---|
| パケット数 | 100 query/s |
| 問合せの応答時間 | 200 msec |
| 攻撃対象の数 | 100 台 |
| キャッシュ有効期間 | 60 sec |
| 使用するポート数 | 40,000 |

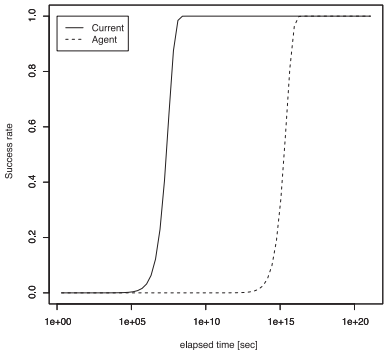



**図 5**　Remote Cache Poisoning 攻撃による成功率の変化
Fig. 5　Success rate of Remote Cache Poisoning Attack.

名前解決方式に関して，それぞれに対する Remote Cache Poisoning 攻撃の成功率を算出した．想定した環境は**表 1** のとおりである．

この環境をもとに，時間経過に対する攻撃成功率をグラフにしたものが**図 5** である．

この結果から，従来の名前解決クライアントは $10^6$ 秒程度，すなわち約 11 日程度で攻撃成功率が 5 割を超えることに対し，本研究で提案した方式の場合は $10^{15}$ 秒程度，つまり約 $3 \times 10^7$ 年ほど攻撃し続けてやっと 2 割を超える程度の攻撃成功率になるということが分かる．

### 4.3　Kaminski Attack に対する評価

Kaminski Attack は特定の DNS キャッシュサーバに意図的に多数の名前解決をさせることにより実行する．通常の DNS キャッシュサーバに意図的に名前解決をさせる方法としては，まずそのサーバに直接問合せを送る方法が考えられる．

しかし本研究で提案した方式では，エージェントは自分自身で DNS ツリーをたどって名前解決を行い，外部からの名前解決を受けないのでこの方法に対しては安全である．

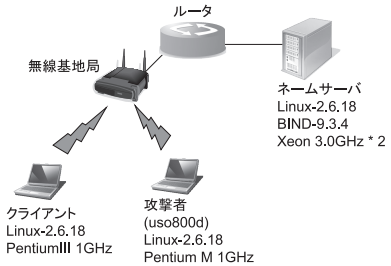



**図 6**　monkey-in-the-middle 攻撃の実験環境
Fig. 6　Environment of monkey-in-the-middle Attack experiment.

それ以外に意図的に名前解決をさせる手法としては，多数の URL を含んだ web ページもしくはメールを送信する方法がある．この場合は DNS キャッシュサーバに直接問合せを送る攻撃方法に比べて攻撃時間は限られており，また本研究の検知システムによりユーザに対して当該 web ページもしくはメールが危険であることを知らせることが可能なため，こちらの攻撃に対しても本研究は有効であるといえる．

また，Kaminski Attack における一番の脅威は，特定ドメインの NS レコードに対応する A レコードが詐称されて既存のキャッシュが上書きされることで発生するドメインの乗っ取り攻撃である．

NS に対する A レコードの攻撃に成功した場合，実装によってキャッシュを上書きするかどうかが異なる．djbdns [6] などの一部の実装はキャッシュの有効期限内であっても A レコードのキャッシュを上書きしてしまうため，この攻撃に対して大きな影響を受ける．

本研究で提案した方式では，キャッシュが存在している場合には NS レコードに対応した A レコードのキャッシュを上書きしないため，この脅威に対しては本方式が通常の Remote Cache Poisoning 攻撃に対する場合と同じ程度の安全性を確保できる．

### 4.4　monkey-in-the-middle 攻撃に対する評価

monkey-in-the-middle 攻撃の実施のために，DNS 攻撃プログラムである uso800d [4] を利用した．**図 6** に実験環境を示す．クライアントおよび攻撃者は同一の無線セグメントに配置し，ルータを介して DNS キャッシュサーバを配置した．実験はあらかじめ用意したドメイン名のリストの順にクライアントが問合せを行い，それに対して攻撃者が monkey-in-the-middle をかけ，その結果を集計した．

**表 2**　monkey-in-the-middle 攻撃実験の結果
Table 2　Result of monkey-in-the-middle Attack experiment.

| | 攻撃によって詐称 | 正しい応答 | タイムアウト | 攻撃を検出 |
|---|---|---|---|---|
| 従来のリゾルバクライアント | 862 | 137 | 1 | - |
| クライアント上でリゾルバクライアントを動作 | 883 | 60 | 57 | - |
| 本研究で提案したエージェント | 27 | 2 | 5 | 966 |

実験に使うドメイン名のリストは，慶應大学湘南藤沢キャンパスにおいて稼動している DNS キャッシュサーバに，ある 1 日に寄せられた DNS 問合せの中より，無作為に 1,000 ドメインを抽出したものを使用した．

実験は以下の 3 つの場合について行った．

(1)　従来のリゾルバクライアントを利用し，DNS キャッシュサーバに対して問合せを行った場合
(2)　クライアント上で DNS キャッシュサーバを動作させ，クライアント自身が再帰的に問合せを行った場合
(3)　クライアント上で本研究が提案したエージェントを動作させ，名前解決を行った場合

正確な実験結果を得るため，各実験を行う際には，毎回 DNS キャッシュサーバ，リゾルバクライアント，および本研究で提案したエージェントの DNS キャッシュをすべて消去してから測定を開始した．

#### 4.4.1　実験結果と考察

**表 2** に実験結果を示す．

従来のリゾルバクライアント，および DNS キャッシュサーバを利用した場合，藤原の研究[3),4)] で示された攻撃の成功率とほぼ変わらない値が観測された．攻撃によって詐称される確率は，自身で再帰呼び出しを行うか否かにはほとんど左右されないことも確認された．

本研究で提案したエージェントは，96%の攻撃を検出した．正しい応答を得られた数が減少しているが，これは名前解決エージェントがサーバからの応答を受信した後にしばらく待機時間をとるという本システムの影響によるものである．正しい応答が先に届いた場合でも，それが正しい応答であると検証する手段がないため，それらすべてを廃棄するからである．

本研究のエージェントを利用した場合でも，27 回の攻撃が成功している．この成功した問合せの内訳は，1 秒を超えてから到達したものが 2 回そもそも回答が戻ってこなかったものが 25 回であった．

DNS は自分が権威を持つゾーンに関しては，存在しないレコードについての問合せであったとしても，即座に応答が帰ってくるように設計されているが，データを保持しているサーバの不調や，誤設定などにより応答が帰ってこない場合がある．そのような場合には攻撃を検出することができない．また，自分が権威を持たないゾーンに関しては，実装によって動作が異なる．再帰問合せを行わないように設定した BIND9[7)] は答えの入っていない NOERROR のパケットを返し，NSD3[8)] は SERVFAIL を返す．djbdns の tinydns はいっさいの返答を返さない．BIND9 および NSD3 は即座に返答が帰ってくるが，djbdns の場合は返答が帰ってこないため攻撃を検出することができない．ただし，権威を持たないゾーンに関する問合せは，ほとんどの場合誤った委譲設定などによって引き起こされるため，通常に起こることではない．

## 5. ま と め

本研究では，従来の DNS プロトコルを利用して DNS 詐称攻撃からクライアントを保護するエージェントの提案を行った．本研究の有効性を検証するために，評価用のエージェントを実装し，実際に攻撃を行うことで評価を行った．その結果，本エージェントは DNS 詐称攻撃に対して効果があることが確認された．本研究によって提案した方式により，DNS 詐称攻撃に対して DNSSEC が利用できない環境においてもクライアントを保護することが可能であることを証明した．

今後の課題としては，まず相手方の DNS サーバが応答をしなかった場合の対策があげられる．相手方のサーバが応答できない状況として，

- 故障やメンテナンスによって先方サーバおよびネットワークが応答できない状態になっている
- サービス不能攻撃などによって先方サーバおよびネットワークが応答できない状態になっている

などが考えられる．これらの攻撃に対処するために，相手方の DNS サーバに対する到達性を確認する，IETF で新たに議論されている DNS cookie を本研究とあわせて利用する，な

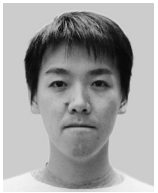
どのさらなる新しい対策が必要となる．

また，アプリケーションに対して攻撃に関する情報を提供する枠組みを作成することがあげられる．現在のアプリケーションが利用する名前解決ライブラリは基本的なホスト名とIP アドレスの相互変換のみの機能に限定されている．そのため，名前解決システムが攻撃を検知したとしてもアプリケーションにその情報を伝えることができない．そこで，セキュリティに関する情報，たとえば署名の有無や攻撃されているかなどをアプリケーションとやりとりできるインタフェースを持った新しい名前解決ライブラリを提案することが考えられる．これによって，ユーザがアプリケーションが行っている名前解決が安全かどうかを確認したり，アプリケーションが自動的にアクセス先の安全性を判断したりすることも可能となり，より安全にシステムを動作させることが可能になると考える．

## 参 考 文 献


1) Vixie, P.: DNS and BIND security issues, *Proc. 5th Conference on USENIX UNIX Security Symposium*, Salt Lake City, Utah, June 05–07, 1995, USENIX Association, Berkeley, CA, Vol.5 (1995).
2) Kaminsky, D.: It's The End Of The Cache As We Know It. http://www.doxpara.com/DMK_BO2K8.ppt
3) Fujiwara, K.: DNS Process-in-the-middle Attack, *ICANN Presentation* (2005).
4) WIDE 合宿における DNS 攻撃実験—Monkey in the middle attack 実験報告，藤原和典，関谷勇司，石原知洋 WIDE プロジェクト 2004 年研究報告，ISSN 1344-9400 (2004).
5) Pettersen, Y.: Enhanced validation of domains for HTTP State Management Cookies using DNS, draft-pettersen-dns-cookie-validate-04, Internet-Draft (Nov. 2008).
6) Bernstein, D.J.: djbdns: Domain Name System tools. http://cr.yp.to/djbdns.html
7) Internet Systems Consortium: ISC BIND. https://www.isc.org/software/bind
8) NLnet Labs.: NSD: Name Server Daemon. http://www.nlnetlabs.nl/projects/nsd/
9) Atkins, D. and Austin, R.: Threat Analysis of the Domain Name System (DNS), RFC3833 (Aug. 2004).
10) Jackson, C., Barth, A., Bortz, A., Shao, W. and Boneh, D.: Protecting Browsers from DNS Rebinding Attacks, *Proc. ACM CCS 07* (2007).
11) Various DNS service implementations generate multiple simultaneous queries for the same resource record, US-CERT Vulnerability Note VU#457875.
12) Mathworld, W.: Birthday Attack. http://mathworld.wolfram.com/BirthdayAttack.html
13) Vixie, P., Gudmundsson, O., Eastlake, D. and Willington, B.: Secret Key Transaction Authentication for DNS (TSIG), RFC2845 (May 2000).
14) Arends, R., Austein, R., Larson, M., Massey D. and Rose, S.: DNS Security Introduction and Requirements, RFC4033 (Mar. 2005).
15) Nah, F.F.-H.: A study on tolerable waiting time: How long are Web users willing to wait?, *Behaviour and Information Technology*, Vol.23, No.3, pp.153–163(11) (2004).
16) Hoxmeier, J.A. and DiCesare, C.: System Response Time and User Satisfaction: An Experimental Study of Browser-based Applications, *Proc. Americas Conference on Information Systems*, 10–13 August 2000, Long Beach, California, Association for Information Systems, pp.140–145 (2000).
17) Jung, J., Sit, E., Balakrishnan, H. and Morris, R.: DNS Performance and the Effectiveness of Caching, *Proc. ACM SIGCOMM Internet Measurement Workshop* (2001).
18) Brownlee, N. and Ziedins, I.: Response time distributions for global name servers, *Proc. PAM 2002 Workshop* (Mar. 2002).


(平成 20 年 6 月 10 日受付)
(平成 20 年 12 月 5 日採録)



**石原　知洋**

1976 年生．2001 年日本大学理工学部物理学科卒業．2003 年慶應義塾大学大学院政策・メディア研究科修士課程修了，同年後期博士課程入学．現在，在籍中．ドメインネームシステム関連の研究・開発に従事．

**関谷　勇司**（正会員）

1997 年京都大学総合人間学部卒業．1999 年慶應義塾大学大学院政策・メディア研究科修了．同年 10 月から 2000 年 3 月まで USC/ISI 訪問研究員として DNS の研究に従事．2005 年慶應義塾大学大学院政策・メディア研究科後期博士課程修了．博士（政策・メディア）．2002 年より東京大学情報基盤センター助手に就任．2007 年同センター助教．次世代ネットワークプロトコルの研究開発と DNS の信頼性向上に関する研究に従事．

**村井　　純**（正会員）

博士（工学）．1979 年慶應義塾大学工学部数理工学科卒業．1981 年同大学大学院理工学研究科修士課程数理工学専攻修了．1987 年 1 月博士（工学）．現在，同大学環境情報学部教授．