

修士論文

DNS Exfiltration 抑止を目的とした分散ハッシュテー
ブルに基づく名前解決システム

高須賀 昌烈

2020 年 3 月 15 日

奈良先端科学技術大学院大学
先端科学技術研究科

本論文は奈良先端科学技術大学院大学先端科学技術研究科に
修士(工学) 授与の要件として提出した修士論文である。

高須賀 昌烈

審査委員：

門林 雄基 教授 (主指導教員)

笠原 正治 教授 (副指導教員)

林 優一 教授 (副指導教員)

妙中 雄三 准教授 (副指導教員)

DNS Exfiltration 抑止を目的とした分散ハッシュテーブルに基づく名前解決システム*

高須賀 昌烈

内容梗概

キーワード

ドメインネームシステム, 情報流出, ネットワークセキュリティ, 分散ハッシュテーブル

*奈良先端科学技術大学院大学 先端科学技術研究科 修士論文, 2020 年 3 月 15 日.

A Name Resolution System to prevent DNS Exfiltration based Distributed Hash Table*

Shoretsu Takasuka

Abstract

Keywords:

Domain Name System(DNS), Information Leakage, Network Security, Distributed Hash Table(DHT)

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, March 15, 2020.

目次

| | |
|------------------------------------|-----------|
| 1. 序論 | 1 |
| 1.1 背景 | 1 |
| 1.2 目的 | 2 |
| 1.3 本論構成 | 2 |
| 2. 準備 | 3 |
| 2.1 DNS プロトコル | 3 |
| 2.1.1 概要 | 3 |
| 2.1.2 定義 | 5 |
| 2.2 DNS Tunneling | 6 |
| 2.2.1 流出メソッド : DNS Exfiltration | 7 |
| 2.2.2 流入メソッド : DNS Infiltration | 8 |
| 3. 関連研究 | 10 |
| 3.1 特徴量に基づく検知手法 | 10 |
| 3.1.1 閾値推定 | 10 |
| 3.1.2 機械学習に基づくモデル | 10 |
| 3.2 新しい DNS アーキテクチャ | 10 |
| 3.2.1 P2P ネットワークを利用した名前解決システム | 10 |
| 3.2.2 Blockchain を利用した名前解決システム | 10 |
| 3.2.3 フラットな名前空間に基づく名前解決システム | 10 |
| 3.3 課題 | 10 |
| 4. 提案システム | 11 |
| 4.1 フラットな名前空間に基づく名前解決システム : REFRES | 11 |
| 4.2 デザイン | 12 |
| 4.2.1 P2P を組み合わせたハイブリッドなアーキテクチャ | 12 |
| 4.2.2 ハッシュ関数に基づくレコード情報の識別子 | 14 |
| 4.2.3 ソートされたハッシュ空間の範囲に基づいたゾーン | 14 |

| | | |
|-----------|-------------------------------------|-----------|
| 4.2.4 | 認証局によるレコード情報の認証基盤 | 16 |
| 4.2.5 | 強く型付けされたレコード情報 | 16 |
| 4.2.6 | レコード情報のデータフォーマット | 16 |
| 4.3 | 動作メカニズム | 16 |
| 4.3.1 | レコード情報に対する操作 | 16 |
| 4.3.2 | 名前解決 | 16 |
| 5. | 評価 | 17 |
| 5.1 | 実装 | 17 |
| 5.2 | 実験環境 | 17 |
| 5.3 | DNS Tunneling に対する理論評価 | 17 |
| 5.3.1 | 評価要素：対 Exfiltration | 17 |
| 5.3.2 | 評価要素：対 Infiltration | 17 |
| 5.4 | シミュレーション実験に基づく特性評価 | 17 |
| 5.4.1 | 評価要素：パケットサイズ | 17 |
| 5.4.2 | 評価要素：トラフィック量 | 17 |
| 5.4.3 | 評価要素：RTT(Round Trip Time) | 17 |
| 6. | 議論 | 18 |
| 6.1 | クエリにおけるハッシュ値計算の最適ノード | 18 |
| 6.2 | 今後の課題 | 18 |
| 7. | 結論 | 19 |
| | 謝辞 | 20 |
| | 参考文献 | 21 |
| | 付録 | 23 |
| A. | 発表リスト (国内研究会) | 23 |

図 目 次

| | | |
|---|--|----|
| 1 | DNS による名前解決 | 5 |
| 2 | ドメインの名前空間 | 6 |
| 3 | arbitrary-string という任意の文字列が, DNS クエリのラベル部を用いて, 事前に用意した権威サーバ (exfil.com) に転送される様子 | 9 |
| 4 | 事前に TXT レコードに登録された情報を問い合わせることで, 権威サーバからの命令情報を取得している様子 | 9 |
| 5 | REFRES の全体図 | 12 |
| 6 | マネージャとプロバイダの関係図 | 13 |

表 目 次

| | | |
|---|---|----|
| 1 | 主要リソースレコード一覧 | 7 |
| 2 | マネージャの情報とそのマネージャが管理するゾーンが記載された対応表 | 15 |

1. 序論

1.1 背景

ドメインネームシステム (Domain Name System, DNS) は、ドメイン名に関連づけられた IP アドレスなどのリソースレコードの情報を解決する機能を担っており、現在のインターネットの利活用において、この仕組みは極めて重要な技術の一つである。1987 年に RFC1034, RFC1035 [1, 2] として公開された DNS のコンセプトは、現在もなお本質的な仕組みは変更されることなく適用されている。他方で、性善説的な当時の設計にはセキュリティやプライバシーが考慮されておらず、DNS には様々な課題が残っており、また設計の脆弱性を悪用した攻撃手法も多数報告されている。その設計に起因する課題の内、DNS クエリのラベルおよびリソースレコード (Resource Record, RR) をデータ転送のメディアとする DNS Tunneling がある。

DNS Tunneling は、一般にフィルタリングされることが少ない DNS の特徴と DNS がデータ転送のメディアとして機能しているとは想像しない人の認知の隙間をついた手法であり、ファイヤー・ウォールや IDS/IPS といったセキュリティラインを突破するために使用される。このように本来の目的とは違う方法でデータを転送する手法は、一般に秘匿通信 (Covert Channel) と呼ばれる [3]。DNS Tunneling は、秘匿通信の代表例であり、マルウェアと C2(Command & Control) サーバとの通信の秘匿手法、または、ターゲットから取得したデータを外部に流出させるといった目的実行の手段として、実際のインシデントで広く利用されている [5, 6, 7, 8, 9, 10, 11, 12]。DNS Tunneling による DNS クエリは、以下 (??) に示すように、転送量に比例して長いラベルを持ち、ラベルとしての文字列制約を満たすためのエンコーディングによって高いエントロピーを示す特徴がある。

また、インタラクティブなシェルなど双方向の通信を DNS Tunneling で実現しようとする場合、時間あたりに高頻度なトラフィックが発生するという特徴が現れる。このような特徴に基づき、パターンマッチングや機械学習、文字列分布などのメソッドを用いた検知手法が過去に多数考案されてきた [13, 14, 15, 16, 17, 18]。それら検知手法は、高い精度で分類を実現しているものがあるが、DNS Tunneling

として検知する対象としているパケットには一般に利用することができる DNS Tunneling ツールキット [19, 20, 21] が使用され、それらは特に過剰な特徴量を示し、明らかに正規の DNS クエリと異なる特徴がある。高い精度を示す従来の検知手法だが、しかし、それらを迂回する手法として、1 回あたりの転送データ量を少なくすることで特徴量を減らす Low Throughput なバイパス手法、また、パケット間のインターバルを数日・数ヶ月と長期化させることでファイル肥大から一定期間しか保存されることがないログ管理の隙間を突いた Slow な Tunneling 手法があり、従来の検知手法では対応することが困難である。悪意を持つユーザの視点として、1bit でも転送できることは秘匿通信として利用することができるため、転送量の少なさは軽視されるべきではない。

他方で、DNS は初めに述べたように、現在のインターネットの根幹技術として根ざしており、抜本的な改変は期待されない。すなわち、既存の DNS による名前解決のメカニズムに大幅な改変を加えないという制約下で、Tunneling に対処することが現実的な最適解であると考えられる。

1.2 目的

本研究では、既存の DNS の名前解決メカニズムの大部分を流用することが一部の改変に留めながら、DNS を用いたデータ転送としての機能の排除を実現する次世代の名前解決メカニズムを提案する。

1.3 本論構成

本稿の構成は次の通りである。第 2 章では、準備として、本論において核となる技術内容・特徴およびそのメカニズムについて説明する。第 3 章では、関連研究としてトラフィックおよびペイロード特徴に基づいた検知手法を説明し、そのバイパス手法として Low Throughput 手法・Slow Tunneling 手法があることを示す。第 4 章では、提案手法を説明する。第 5 章では、提案手法の DNS Tunneling に対する評価を行い、併せて提案手法の特定についても説明する。第 6 章では、提案手法の課題について議論する。最後に、第 7 章にて結論を述べる。

2. 準備

本章では、本論において核となる技術内容・特徴およびそのメカニズムについて説明する。

2.1 DNS プロトコル

2.1.1 概要

本節では、DNS(Domain Name System)の概要について説明する。DNSは、インターネットに接続された無数のコンピュータを一意に識別するためのIPアドレスを、人が認識しやすいドメイン名に変換するシステムである。元来、インターネット上でのホストの識別にはIPアドレスが使用されてきた。しかし、32bitの名前空間で10進数表記のIPv4(E.g. “192.168.0.1”)や128bitの名前空間で16進数表記のIPv6(E.g. “2001:200:16a:8::230”)は、決して人にとって認識しやすいものではない。そこで、自然言語のようにアルファベットや数字で表記されるホスト名とIPアドレスを関連づける仕組みが登場した。当初、hosts.txtと呼ばれる対応表は、中央集権的に管理されていた。しかし、ホスト数の増大に伴い、管理が困難になっていく。その解決策として登場したのが、管理するドメインをゾーンで切り出し権限を委譲していく分散的な階層構造を持つDNSである。

DNSのシステムアーキテクチャは、クライアント・サーバ構成で成り立っている。一般に、クライアントがドメインを問い合わせた場合、サーバはドメインに対応づけているIPアドレスを応答することで、クライアントはドメインに対応づけられたIPアドレスを解決することができる。DNSのサービスは、機能に応じて3つに分けられる。

- Stub Resolver
- Full Service Resolver
- Authorization Server

スタブリゾルバは、名前解決の問い合わせを依頼するクライアントノードである。フルサービスリゾルバ(キャッシュサーバ・リカーシブサーバとも呼称され

る) は、スタブリゾルバからの名前解決問い合わせをハンドリングするノードである。過去に問い合わせられた情報をキャッシュとして保持する機能と、レコード情報を保持・提供する権威サーバに対する再帰問い合わせを行う機能を担う。一般に、フルサービスリゾルバは、“root.hints”というルート権威サーバとそのアドレスが対応づけられたファイルを保持しており、再帰問い合わせの際にはこのファイルに基づき、最初の宛先となる権威サーバのアドレスを解決する。権威サーバは、レコード情報を保持するサーバノードであり、フルリゾルバからの転送される問い合わせ依頼に応答する。

クライアントから“www.example.com”のIPv4アドレスについて問い合わせられた場合を考える。はじめに、クライアントであるスタブリゾルバは、スタブリゾルバと同一セグメント内のフルサービスリゾルバもしくは、ネットワークセグメントに依らずインターネット上のどのクライアントからもアクセスできるパブリックなフルサービスリゾルバ(オープンリゾルバ、パブリックリゾルバとも呼称される)に問い合わせる。フルサービスリゾルバははじめに、ドメイン名が“www.example.com”で、リソースレコードが“A”の応答レコードがキャッシュにあるかどうかを判別する。キャッシュにヒットした場合にはキャッシュの情報をクライアントに応答され、ヒットしなかった場合には、root.hints ファイルを参照しルート権威サーバにリクエスト packets を転送する。クエリ(問い合わせ)を受け取ったルート権威サーバは、ルートゾーン内の“com”ドメインが委譲された権威サーバのアドレスを応答する。次に、フルサービスリゾルバは、“com”ドメインが委譲された権威サーバに対し同様のクエリを転送する。“com”ドメインを管理する権威サーバは、同様にして、“com”ゾーン内の“example.com”が委譲された権威サーバのアドレスを応答する。フルリゾルバは、“example.com”ドメインを委譲された権威サーバ宛に同様のクエリを転送する。“example.com”ゾーンを管理する権威サーバは、保持するゾーンファイルからクエリされたドメインのリソースレコードについて探索し、探索の結果としてレコード情報をフルサービスリゾルバに応答する。フルサービスリゾルバは、権威サーバからの応答されたレコード情報の TTL(Time To Live) の期間レコード情報をキャッシュしたのち、クライアントのスタブリゾルバに結果を応答する。このようにして、再帰的な問

い合わせの仕組みに基づき名前解決が行われる。

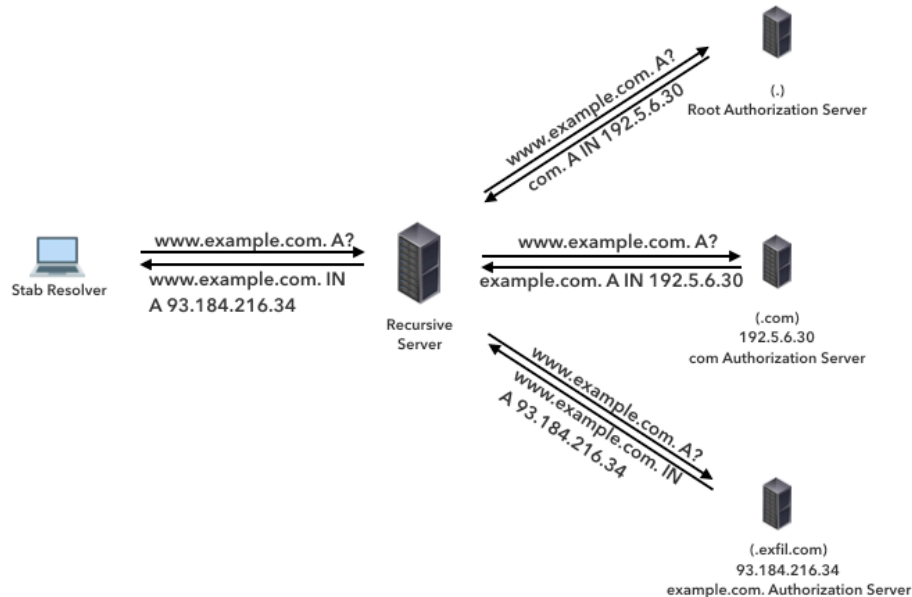


図 1 DNS による名前解決

2.1.2 定義

本節では、DNS における用語と概念について説明する。

ドメインは、数字とアルファベットおよびハイフン (“-”) の文字列で表記され、最大長は 63 オクテットと定義される。また、ドメインはルートを頂点とする階層構造で構成され、各階層にはドメインを管理する主体として権威サーバが存在し、管理主体を委譲していくことによって分散的にデータベースを管理する仕組みで動作する。

ドメイン名は、ドメインに相当するラベルをドット区切りで表され、最大長は 255 オクテットである。ドメイン名は右から順に階層序列が表現され、ドットで表現されるルートは一般には省略される。最も右に位置づくラベルが TLD(Top Level Domain) であり、その TLD から n 番目 ($n \mid n \in \mathbb{N}$) のラベルが第 n レベルドメインである。

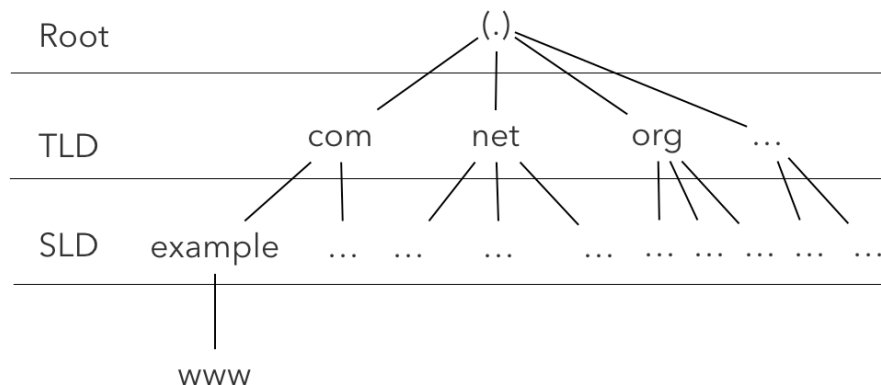


図 2 ドメインの名前空間

TLDを大別すると，“.com”や“.net”をはじめとした特定分野別の gTLD(global Top Level Domain), “.jp”や“.ch”のような国ごとに割り当てられている ccTLD(Country Code Top Level Domain) の二つに分けられる。

ドメイン名に関連づけられる情報は，リソースレコード (Resource Record, RR) と呼ばれる。リソースレコードは，タイプが定義されており，目的ごとに使用されるタイプが異なる。最も一般的なレコード，A レコードタイプは，ドメインに対して IPv4 アドレスを関連づけるために使用される。名前解決において，クライアントはドメイン名とそのドメインに関連づけられたリソースレコードを指定する。これによって，クライアントは，ドメインに関連づけられたリソースレコード情報を取得することができる。表 1 は，主要なリソースレコードである。

2.2 DNS Tunneling

本節では，DNS Tunneling について説明する。DNS Tunneling は，DNS をデータ転送のメディアとした秘匿通信手法の総称である。第 2.1 節で示す通り，DNS は，ドメイン名とそのドメイン名に関連づけられたリソースレコードのレコード情報を解決するシステムであり，現代のインターネットにおいて極めて重要な役割を担うネットワークプロトコルである。このため，DNS はトラフィックがフィルタリング・モニタリングされにくいプロトコルの一つでもある。しかし，DNS のデザインは，クライアントサーバアーキテクチャに基づき，クライアントのス

| タイプ | 値 | 目的 |
|-------|----|--------------------|
| A | 1 | ホストの IPv4 アドレス |
| NS | 2 | 権威サーバ |
| MF | 4 | メール転送サーバ |
| CNAME | 5 | 別名 |
| SOA | 6 | 権威ゾーンの開始 |
| NULL | 10 | NULL(実験用) |
| PTR | 12 | ドメイン名のポインター (逆引き) |
| HINFO | 13 | ホスト情報 |
| MINFO | 14 | メールボックスおよびメールリスト情報 |
| MX | 15 | メール交換 |
| TXT | 16 | 任意文字列 |

表 1 主要リソースレコード一覧

タブリゾルバと権威サーバが直接やり取りされるため、の問い合わせ情報が権威サーバに転送されるため、転送される仕組みは、データ転送の仕組みでもある。

2.2.1 流出メソッド : DNS Exfiltration

DNS Exfiltration は、DNS のクエリが DNS の再帰問い合わせの仕組みに基づいて権威サーバに到達することを利用することで、クライアントから権威サーバ方向に対してデータを転送する手法である。一般に、組織のネットワークにセキュリティ対策を講じる時、DNS の 53 ポートを制限することは極端に利便性を阻害することからフィルタリング処理が施されることは少ないとされる。DNS Exfiltration は、上記の DNS の特性を悪用し、悪意の対象となるネットワークなどの外部との通信が制限されている環境において、外部にデータを転送する際に効果を発揮する。過去のインシデントでは、権威サーバをデータ回収のサーバとして利用するケースや、次節 2.2.2 の流入通信とを組み合わせ、命令サーバとの相互通信の流出方向の通信手段として利用されてきた。この手法では、クエリを

送信する主体とそのクエリを受信する権威サーバを実行者の制御下にあることを想定している。すなわち、Exfiltrationを実施するには、予めデータの宛先となる権威サーバを動作させるために、グローバルなドメイン名を用意する必要がある。例えば、いま、“exfil.com”というドメインを保持する権威サーバを考える。この権威サーバは、“exfil.com”以下の全ての名前空間をゾーンとして管理することができる。データを転送する際のキャリアとなるのは、DNS クエリの“exfil.com”以下のラベルである。DNS のラベルは、数字・アルファベット・先頭以外でハイフン (“-”) という文字列の制約があるため、任意の文字列をそのまま注入することは困難である。そこで、ASCII コードに変換することが可能な Base32・Base64 などの文字列エンコーディングを適用させることで、文字列の制約条件を満たす。任意の文字列を文字列エンコーディングの手法で変換させ、用意できた文字列を QNAME に含め、適当なりソースレコードタイプを指定すれば、宛先となる権威サーバにデータが転送される。最後に、ラベルを同一の文字列エンコーディングアルゴリズムでデコードすることで、元のデータを受け取ることができるという具合である。このようにして、DNS の名前解決の仕組みを応用することで、任意の権威サーバに任意の文字列を転送することができる。これが DNS Exfiltration の動作メカニズムである。図 3 に、DNS Exfiltration のメカニズムを図解した様子である。

2.2.2 流入メソッド : DNS Infiltration

DNS Infiltration は、管理する権威サーバのドメインに適当なホスト名 (E.g. www) を作成し、そのホスト名のリソースレコード (E.g. TXT) に情報を付与していた場合には、そのホストへの問い合わせを通じて逆方向、すなわち権威サーバから任意の情報を転送することができる。DNS のリソースレコードを転送キャリアとする流入通信のメカニズムを図解した様子が、図 4 である。

このようにして、Exfiltration と Infiltration メソッドを使うことで、データを送受信することができる。

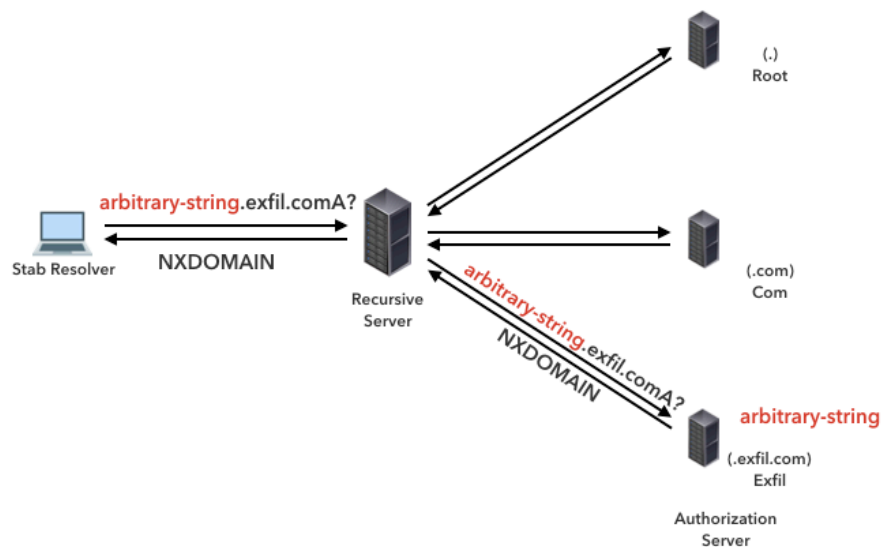


図 3 arbitrary-string という任意の文字列が、DNS クエリのラベル部を用いて、事前に用意した権威サーバ (exfil.com) に転送される様子

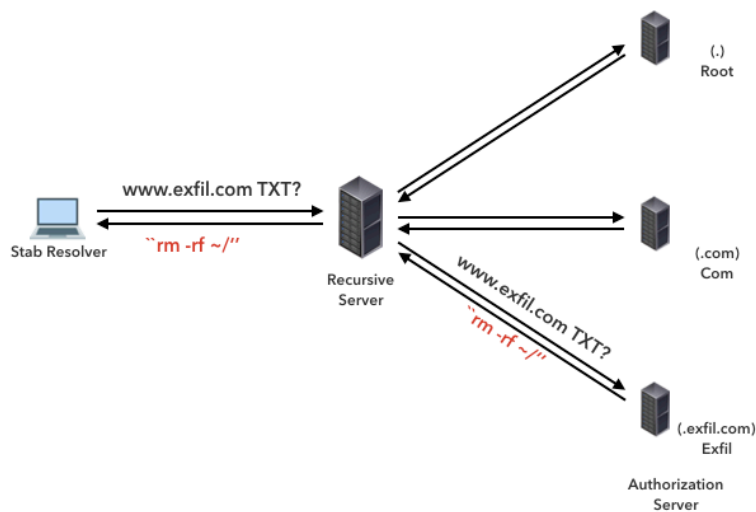


図 4 事前に TXT レコードに登録された情報を問い合わせることで、権威サーバからの命令情報を取得している様子

3. 関連研究

本章では，はじめに既存の DNS Tunneling に対するアプローチとして提案されている検知アプローチを取り上げ，現在の検知に基づく対策の課題として，Low Throughput 手法と Slow な転送手法というバイパス手法に対処できないことを明らかにする．次に，これまでに提案されてきた P2P ベースの名前解決システムを説明し，提案手法との違いを示す．

3.1 特徴量に基づく検知手法

3.1.1 閾値推定

3.1.2 機械学習に基づくモデル

DNS Tunneling メソッドを使用した時の DNS クエリは，??で述べたような特徴が現れる性質がある．この性質に基づき，これまでに多数の検知手法が提案されてきた．

Born ら [13] は，

3.2 新しい DNS アーキテクチャ

3.2.1 P2P ネットワークを利用した名前解決システム

これまでに，DNS における～の課題に対して，P2P に基づいた名前解決システムは数多く提案されてきた．

3.2.2 Blockchain を利用した名前解決システム

3.2.3 フラットな名前空間に基づく名前解決システム

3.3 課題

4. 提案システム

本章では、提案システムについて説明する．はじめにシステムの概要と目的を示す．次に、システムの詳細に関して、アーキテクチャ・動作メカニズム・プロトコル・スケーラビリティを順に説明する．

4.1 フラットな名前空間に基づく名前解決システム：REFRES

本節では、システムの概要について説明する．

REFRES(REcorded inFormation REsolution System) は、フラットな名前空間においてハッシュ値の範囲に基づきゾーンを分割し、そのゾーンを管理する主体として既存システムの TLD を割り当てることで、名前解決の手続きにおけるノードをクライアントとゾーン管理ノードのみに限定させた名前解決システムである．REFRES では、ドメイン名 “www.example.com” の A レコードやドメイン名 “www.example.com” の TXT レコードなど全てのレコード情報をフラットな名前空間上で管理する．また、ドメイン名とリソースレコードタイプもしくは IP アドレスとリソースレコードタイプから算出されるハッシュ値を識別子とすることでレコード情報へのアクセスを実現する．このようにして、REFRES におけるクライアントは、レコード情報を保持するノードを一意に特定することができる．具体的なレコード情報は、ゾーンを管理するサーバノードに階層的に連結したノードのよって作成・更新・破棄される．レコード情報を操作する際には、事前に認証局との認証手続きをすることで、レコード情報の信頼を保証する．最終的に REFRES では、クライアントからの名前解決リクエストと偽装して送信される特定の権威サーバ指向の悪性クエリが発生する仕組みを抑止する．REFRES の特徴を以下に示す．

- フラットな名前空間
- ハッシュ値の範囲に基づくゾーニング
- レコード情報の操作と提供・管理の機能分離
- レコード情報に対する識別子の導入
- 全てのレコード情報が認証済み

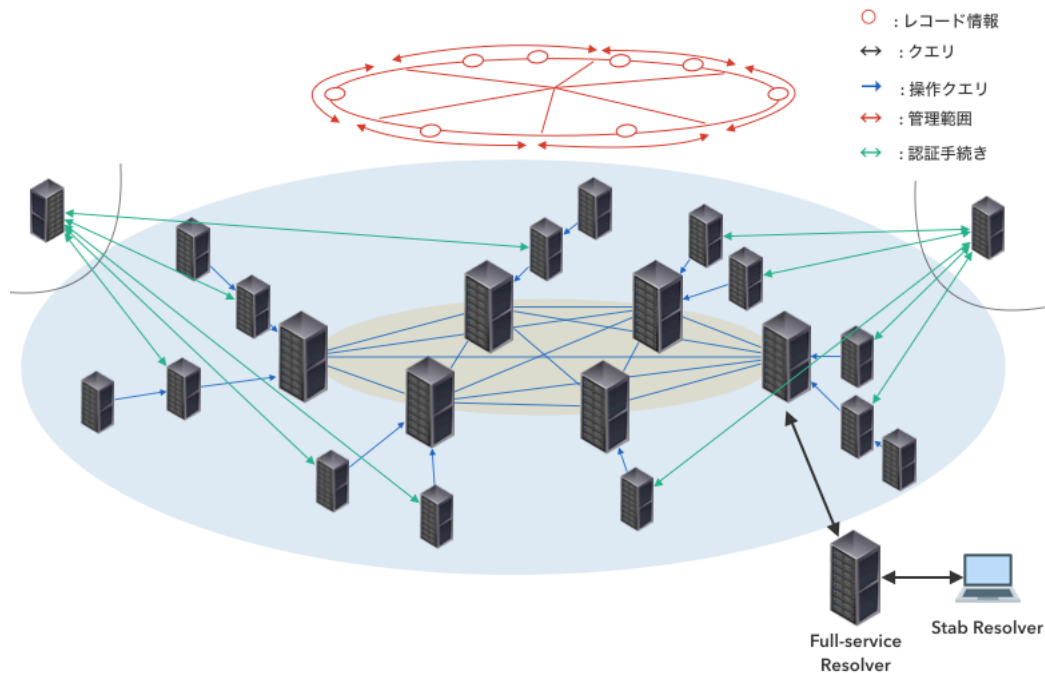


図 5 REFRES の全体図

4.2 デザイン

4.2.1 P2P を組み合わせたハイブリッドなアーキテクチャ

全体のアーキテクチャに関して、REFRES では、既存の DNS 同様にクライアントサーバアーキテクチャで構成される。サーバは、フルメッシュネットワークで相互に接続される固定ノードによる P2P ネットワークアーキテクチャで構成される。

REFRES におけるサービスノードは、機能に基づいて 4 つに分けることができる。スタブリゾルバは、既存の DNS と同様に動作するノードであり、DNS クライアントとして名前解決を依頼する主体として位置づくノードである。フルサービスリゾルバもスタブリゾルバ同様、1 つ追加機能を除いて変更点はなく、スタブリゾルバからの問い合わせに対してリソース情報を保持する主体に代理的に問い合わせ機能と、問い合わせた情報を一定期間キャッシュするキャッシュサーバとして機能するノードである。変更された点は、以降で説明するように、REFRES

ではレコード情報に対する識別子をキーとしてアクセスする．フルサービスリゾルバは，スタブリゾルバからの従来のフォーマットの DNS クエリについて，ドメイン名とレコードタイプからハッシュ値の算出を行う機能を担う．算出したハッシュ値をキーとして，フルサービスリゾルバは，サーバに転送する．REFRES では，権威サーバが機能に基づき二つサービスノードに分割される．

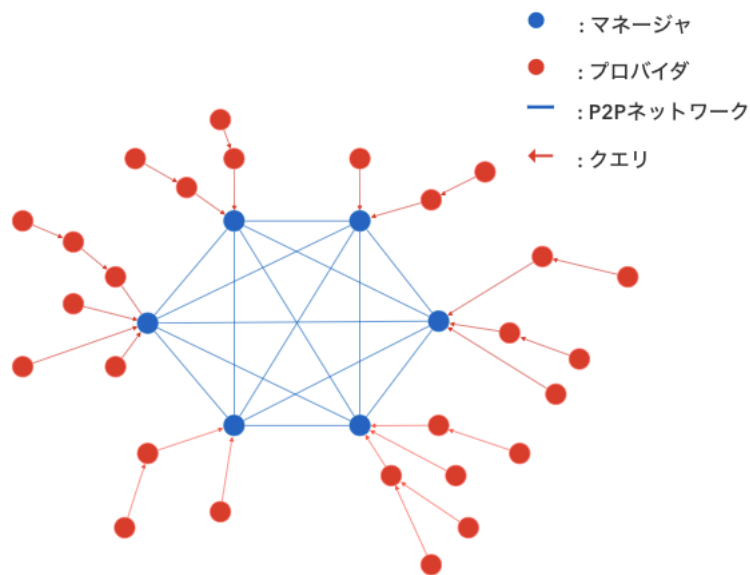


図 6 マネージャとプロバイダの関係図

マネージャは，実際のレコード情報を保持し，クライアントからの問い合わせに応答するサーバノードである．また，マネージャは，既存の DNS における TLD に位置づく権威サーバが担当する．続いて，4つ目のサービスノードとなるプロバイダは，レコード情報を作成・更新および消去といった操作を担当する．プロバイダは，既存の DNS における SLD 以降の権威サーバに相当し，ドメインの階層構造を従いマネージャに接続される．プロバイダは，マネージャを介在することで，レコード情報を操作することができる．例えば，example.com プロバイダが“www”の IP アドレス情報を作成することを考える．example.com プロバイダは，“www.example.com”とレコードタイプ“A”およびその値“93.184.216.34”を含むデータを接続先の com マネージャにリクエストする．com マネージャは，リクエストされたドメイン名とそれに関連づけるレコードタイプから識別子を算出

し、担当のマネージャにストアリクエストを転送するという具合で動作する。

4.2.2 ハッシュ関数に基づくレコード情報の識別子

本節では、レコード情報に付与する識別子について説明する。正引きの場合、従来のシステムでは、“ドメイン名”と“レコードタイプ”の二つの情報をキーとして、サーバは保持するゾーンファイルから該当するレコード情報の値が求まる。REFRES では、“ドメイン名”と“レコードタイプ”を表す文字列の和を引数として算出されるハッシュ値を識別子として利用することで、その識別子をキーを持つデータが求まる。ドメイン名とそれに関連づけられたレコード情報の組み合わせについて、それら要素の文字列の和を引数としてハッシュ関数に適用することで、算出されるダイジェストを識別子として機能する構成となっている。

例えば、あるドメイン名に関連づけられたレコードタイプ A のレコード情報について取得する場合を考える。そのレコード情報が“com”サーバによって管理されているとき、クライアントは“com”サーバに対して識別子を含むリクエストパケットを転送することで、“com”サーバは識別子に対応するレコード情報をクライアントに応答するという具合である。

次に、識別子を算出する際に使用するハッシュ関数のアルゴリズムについて説明する。(パケットフォーマット制約について説明) DNS のパケットフォーマットは、5つのセクションに分類される。(衝突強度について説明) 以上より、REFRES ではパケットフォーマットに対する出力長と衝突に対する強度に基づき、224bit の名前空間をもつ sha3 アルゴリズムをハッシュ関数アルゴリズムとして用いる。

4.2.3 ソートされたハッシュ空間の範囲に基づいたゾーン

本節では、ゾーンの分割方法およびマネージャノードのアドレスとそのゾーンの範囲に関する対応表について説明する。はじめに比較のために、従来のシステムの場合について説明する。従来のシステムでは、ドメインの階層構造に従い、ドメインの管理ノードを下位のドメイン管理ノードに委譲することでゾーンが分割される。この仕組みでは、ゾーン内の全てのレコード情報はゾーンファイルに

画一的にまとめられ、そのゾーンを管理する権威サーバがレコード情報の保持機能とクライアントから応答するという二つの機能を担う。このゾーン分割メソッドでは、レコード情報の帰属が明確であり、ドメインの管理ノードがトラストアンカーとしての役割を同時に担うことができるメリットがある。一方の REFRES では、識別子を算出する際に使用するハッシュ関数によって構成される名前空間に基づき、ソートされたハッシュの名前空間の連続した範囲で分割する。この分割された連続した範囲に基づきゾーンがマネージャに割り当てられることで、既存システム同様にレコード情報全体を分散的に管理する。

上記で説明するように、マネージャが管理するゾーンは、ハッシュの名前空間の連続した一部の範囲である。従って、レコード情報は、ハッシュの名前空間上で識別子をソートした際に、帰属する範囲を管理するマネージャによって保持される。マネージャのアドレスを解決する方法には、ゾーンとしてハッシュ値の範囲とそのマネージャおよびマネージャのアドレスに関する対応表 2 によって解決される。REFRES では、フルサービスリゾルバ・マネージャ・プロバイダのノードがこの対応表を保持することで、識別子から一意にマネージャとそのマネージャのアドレスを特定する。は、どのマネージャがどのハッシュ空間をゾーンとして管理しているのかに関する対応表を保持している。

| ゾーン | マネージャアドレス | ホスト名 |
|--------------------|---------------|-------|
| (00...00, 4z...zz) | 192.35.51.30 | com. |
| (50...00, az...zz) | 192.5.6.30 | net. |
| (b0...00, gz...zz) | 199.249.112.1 | org. |
| (h0...00, mz...zz) | 213.248.216.1 | uk. |
| (n0...00, sz...zz) | 199.254.31.1 | info. |
| (t0...00, zz...zz) | 194.0.0.53 | de. |

表 2 マネージャの情報とそのマネージャが管理するゾーンが記載された対応表

4.2.4 認証局によるレコード情報の認証基盤

本節では、レコード情報の認証基盤について説明する。REFRES では、レコード情報の信頼性を担保するために、信頼できる第三者機関との認証プロセスをレコード情報のストアリングフェーズに介在させる。REFRES におけるレコード情報は、その情報の信頼性を担保するために、信頼できる第三者機関から発行される証明書を REFRES では、不審なレコード情報がハッシュテーブルにストアさせることを防ぐために、認証局モデルを採用する。ために、全てのレコード情報をストアする前に認証局での評価を介在させる。レコード情報をハッシュテーブル上にストアする前に認証局を介在させ、ハッシュテーブル上にストアされるレコード情報に関して、認証機関を介在させることによって、プロバイダの真正性と不審なレコード情報をふるいにかける。

4.2.5 強く型付けされたレコード情報

本節では、REFRES で使用されるリソースレコードのタイプについて説明する。第 2.2.2 で示すように、既存の DNS で使用されるリソースレコードには、事前に特定のリソースレコードに対して任意の文字列を登録しておくことで外部からデータ転送のキャリアとして機能する特性がある。本節では、DNS Infiltration の対策を主たる目的として再構築したリソースレコードについて説明する。

4.2.6 レコード情報のデータフォーマット

4.3 動作メカニズム

4.3.1 レコード情報に対する操作

DNS Infiltration に対応するため、リソースレコードを再定義する。本節では、REFRES において使用されるリソースレコードのタイプと

4.3.2 名前解決

5. 評価

5.1 実装

5.2 実験環境

5.3 DNS Tunneling に対する理論評価

5.3.1 評価要素：対 Exfiltration

5.3.2 評価要素：対 Infiltration

5.4 シミュレーション実験に基づく特性評価

5.4.1 評価要素：パケットサイズ

5.4.2 評価要素：トラフィック量

5.4.3 評価要素：RTT(Round Trip Time)

6. 議論

6.1 クエリにおけるハッシュ値計算の最適ノード

6.2 今後の課題

7. 結論

謝辞

ご指導ご鞭撻賜りありがとうございました.

参考文献

- [1] P.V. Mockapetris. “Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD),” November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.
- [2] P.V. Mockapetris. “Domain names - implementation and specification. RFC 1035 (INTERNET STANDARD),” November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.”
- [3] ICANN, “What Is an Internet Covert Channel?,” August 2016. <https://www.icann.org/news/blog/what-is-an-internet-covert-channel>. (accessed 2019-12-12).
- [4] S. Bortzmeyer. “DNS Privacy Considerations RFC 7626 (INTERNET STANDARD), ” August 2015.
- [5] KrebsOnSecurity. “Deconstructing the 2014 Sally Beauty Breach,” May 2015. <https://krebsonsecurity.com/2015/05/deconstructing-the-2014-sally-beauty-breach/>. (accessed 2019-11-30).
- [6] IronNet. “Chirp of the PoisonFrog,” February 2019. <https://ironnet.com/blog/chirp-of-the-poisonfrog/>. (accessed 2019-11-30).
- [7] Nick Hoffman. “BernhardPOS,” July 2015. <https://securitykitten.github.io/2015/07/14/bernhardpos.html>. (accessed 2019-11-30).
- [8] Fireeye. “MULTIGRAIN – Point of Sale Attackers Make an Unhealthy Addition to the Pantry,” April 2016. https://www.fireeye.com/blog/threat-research/2016/04/multigrain_pointo.html. (accessed 2019-11-30).
- [9] Palo alto Networks. “New Wekby Attacks Use DNS Requests As Command and Control Mechanism,” May 2016. <https://unit42.paloaltonetworks.com/unit42-new-wekby-attacks-use-dns-requests-as-command-and-control-mechanism/>. (accessed 2019-11-30).
- [10] Kaspersky. “Use of DNS Tunneling for C&C Communications,” April 2017. <https://securelist.com/use-of-dns-tunneling-for-cc-communications/78203/>. (accessed 2019-11-30).

- [11] CISCO Talos. “Spoofed SEC Emails Distribute Evolved DNSMessenger,” October 2017. <https://blog.talosintelligence.com/2017/10/dnsmessenger-sec-campaign.html>. (accessed 2019-11-30).
- [12] Cylance. “Threat Spotlight: Inside UDPOs Malware,” February 27 2018. https://threatvector.cylance.com/en_us/home/threat-spotlight-inside-udpos-malware.html. (accessed 2019-11-30).
- [13] K. Born and D. Gustafson, “NgViz: detecting DNS tunnels through n-gram visualization and quantitative analysis,” Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tennessee, 2010, pp. 1-4.
- [14] Cheng Qi, Xiaojun Chen, Cui Xu, Jinqiao Shi, Peipeng Liu, “A Bigram based Real Time DNS Tunnel Detection Approach,” Procedia Computer Science, Volume 17, 2013, Pages 852-860.
- [15] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang and C. Peng, “Detecting DNS Tunnel through Binary-Classification Based on Behavior Features,” 2017 IEEE Trustcom/BigDataSE/ICCESS, Sydney, NSW, 2017, pp. 339-346.
- [16] Asaf Nadler, Avi Aminov, Asaf Shabtai, “Detection of malicious and low throughput data exfiltration over the DNS protocol,” Computers & Security, Volume 80, 2019, Pages 36-53.
- [17] J. Steadman and S. Scott-Hayward, “DNSxD: Detecting Data Exfiltration Over DNS,” 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-6.
- [18] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell and V. Sivaraman, “Monitoring Enterprise DNS Queries for Detecting Data Exfiltration from Internal Hosts,” in IEEE Transactions on Network and Service Management.
- [19] “OzymanDNS - Tunneling SSH over DNS,” <https://room362.com/post/2009/2009310ozymandns-tunneling-ssh-over-dns.html/>, (accessed 2019-11-20).
- [20] “iodine,” <http://code.kryo.se/iodine/>, (accessed 2019-11-20).
- [21] “DNSScat2,” <https://github.com/iagox86/dnscat2>, (accessed 2019-11-20).

付録

A. 発表リスト (国内研究会)

1. 高須賀 昌烈, 妙中 雄三, 門林 雄基, “非実在ドメインに対するネガティブキャッシュの拡張と再帰問い合わせハッシュ化の提案”, 電子情報通信学会情報ネットワーク研究会, 2019-10-ICTSSL-IN, 2019 年 10 月.