修士論文

DNS Tunneling抑止を目的としたシンボル志向型名前解決システム

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 Tunneling抑止を目的としたシンボル志向型名前解決システム*

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，分散ハッシュテーブル，P2P ネットワーク，DNS Tunneling，ネットワークセキュリティ

*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 Symbok Oriented Name Resolution System to prevent DNS Tunneling*

Shoretsu Takasuka

Abstract

Keywords:

Domain Name System(DNS), Distributed Hash Table(DHT), P2P Network, DNS Tunneling, ネットワークセキュリティ

*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 構造 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 9

3. 関連研究 **11**

3.1 特徴量に基づく検知手法 11

3.1.1 閾値推定 11

3.1.2 機械学習に基づくモデル 11

3.2 ポスト名前解決システム 11

3.2.1 P2P ネットワークを利用した名前解決システム 11

3.2.2 Blockchain を利用した名前解決システム 12

3.3 課題 12

4. 提案システム **13**

4.1 フラットな名前空間に基づく名前解決システム：REFRES 13

4.2 デザイン 14

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 . . . 14

4.2.2 ハッシュ関数に基づくレコード情報の識別子とハッシュアルゴリズム 16

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 . . . 18

4.2.4 認証基盤に基づく信頼されるレコード情報 19
4.2.5 強い型付けのレコード情報による DNS Infiltration 20
4.2.6 コンテンツのデータフォーマット 22
4.3 動作メカニズム 22
4.3.1 レコード情報に対する操作 22
4.3.2 名前解決 22

5. 評価 **23**
5.1 実装 23
5.2 実験環境 23
5.3 DNS Tunneling に対する理論評価 23
5.3.1 評価要素 : 対 Exfiltration 23
5.3.2 評価要素 : 対 Infiltration 23
5.4 シミュレーション実験に基づく特性評価 23
5.4.1 評価要素 : パケットサイズ 23
5.4.2 評価要素 : RTT(Round Trip Time) 23
5.4.3 評価要素 : トラフィック量 23

6. 議論 **25**
6.1 クエリにおけるハッシュ値計算の最適ノード 25
6.2 今後の課題 25

7. 結論 **26**

謝辞 **27**

参考文献 **28**

付録 **31**

A. 発表リスト (国内研究会) **31**

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 8
4 TXT レコードに登録された情報について，DNS クエリで問い合わせることで権威サーバから命令情報を取得している様子 10
5 REFRES の全体図 14
6 マネージャとプロバイダの関係図 15
7 レコード情報に対する認証プロセス 20
8 コンテンツファイルのデータフォーマット 22

表 目 次

1 主要リソースレコード一覧 7
2 REFRES における用語 16
3 6 つのマネージャによって管理されるハッシュテーブルにおいて，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の例 19
4 DNS Infiltration として利用することができるリソースレコード一覧 21

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 [4]．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 [6, 7, 8, 9, 10, 11, 12, 13]．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 [14, 15, 16, 17, 18, 19]．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 [20, 21, 22] 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 Tunenling 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 の名前空間をもつ 10 進数表記の IPv4(E.g. “192.168.0.1”) 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．

- スタブリゾルバ
- フルサービスリゾルバ
- 権威サーバ

スタブリゾルバは，名前解決の問い合わせる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パケットを転送する．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とり行われる．

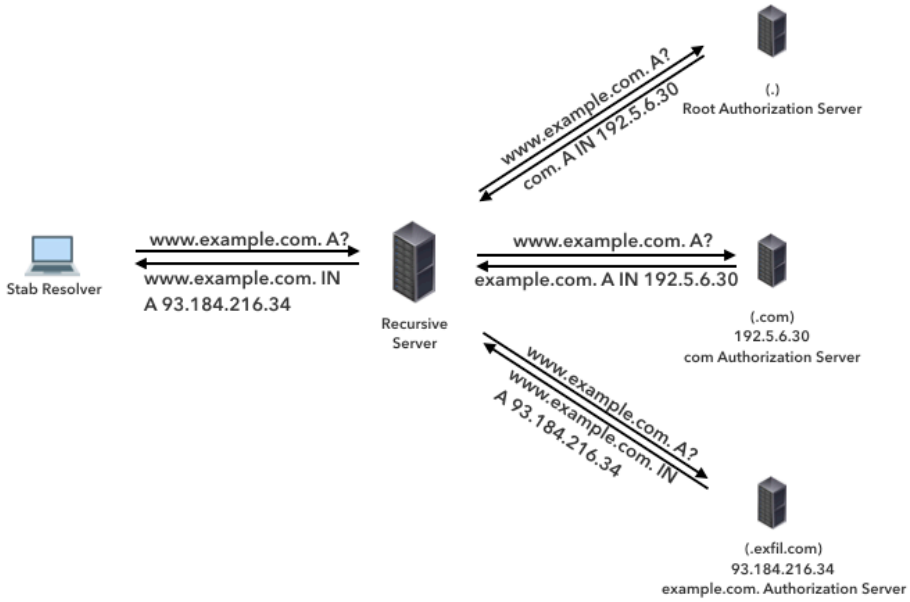

図 1 DNS による名前解決

2.1.2 構造

本節では，DNS における構造について説明する．

DNS は，階層ごとに名前空間を分割させた識別子の分散管理システムである．このシステムは，ルートを頂点として，その下層に独自の名前空間を保持するドメインをツリー状に連結されていく構造をとる．各ドメインは，権威サーバによってドメインの名前空間が管理される．ドメインの名前空間はゾーンと呼ばれ，そのドメインに位置づく権威サーバによって管理される．DNS には委譲の仕組みがあり，権威サーバは自身が管理するゾーンについてドメインが枝分かれしていっている場合，下層の権威サーバにゾーンを管理を譲ることができる．一般にルートの直下で構成されるドメインを TLD(Top Level Domain) と言い，“com.”や “net.”などがある．さらにその下層には，SLD(Second Level Domain) が続き，TLD から n 番目 $(n \mid n \in \mathbb{N})$ のラベルが第 n レベルドメインと序列していく具合である．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(Co
untry Code Top Level Domain) の二つに分けられる．

ドメイン名は，ドット区切りでラベルを連結形式で表記され，一般にルートを意味する最も右のドットは省略される．ドメイン名は，ルートと区切り文字を除いて最大が 253bit である．ラベルは，各ドメインを表し，数字とアルファベットおよびハイフン (“-”) の文字列から表記される最大長 63bit で定義される．

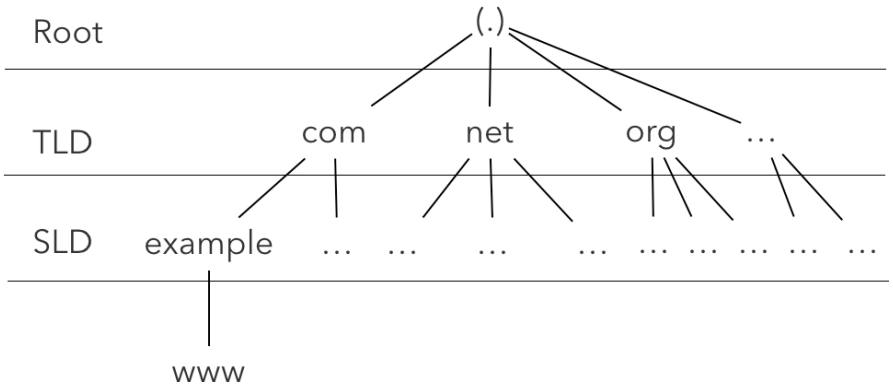

図 2 ドメインの名前空間

ドメイン名に関連づけられる情報は，リソースレコード (Resorce 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

タイプ	値	意味
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

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て，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

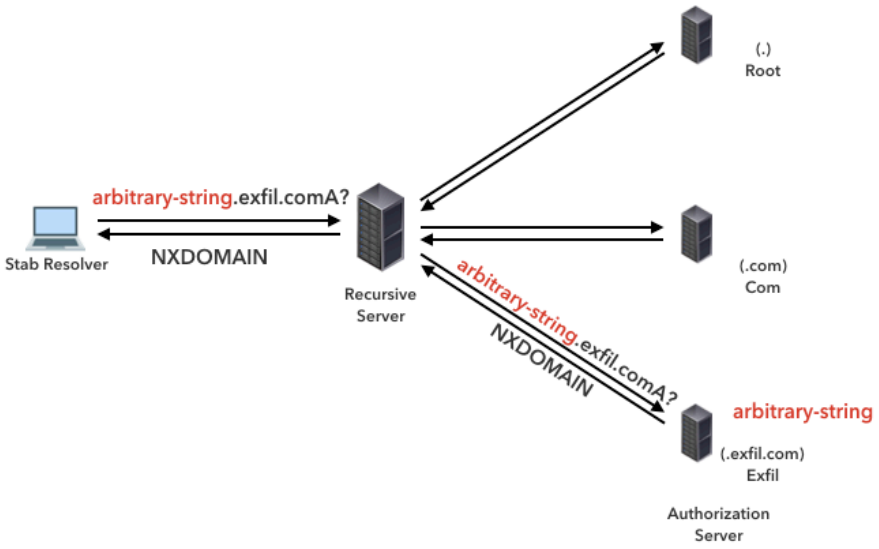

図 3　arbitrary-string という任意の文字列が，DNS クエリのラベル部を用いて，事前に用意した権威サーバ (exfil.com) に転送される様子

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，適当なリ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，DNS Exfiltration と逆方向の権威サーバからクライアント方向にデータを転送する手法である．この手法は，DNS のリソースレコードに任意の文字列が含められる設計になっていることを利用したものである．リソースレコードのタイプは多種多様であるが，ゾーンファイルを自由に編集できる現在の DNS エコシステムでは，ドメイン名との関わりに関係なく任意の情報をドメイン名に関連づけることが可能である．DNS Infiltration を実施するには，送信元となる権威サーバのドメイン名に，適当なリソースレコードのタイプ (E.g. TXT) に転送したい文字列を登録しておくことで，そのドメイン名を QNAME としてそのレコードタイプを問い合わせることで，クライアントのもとでデータを回収することができるという具合である．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．

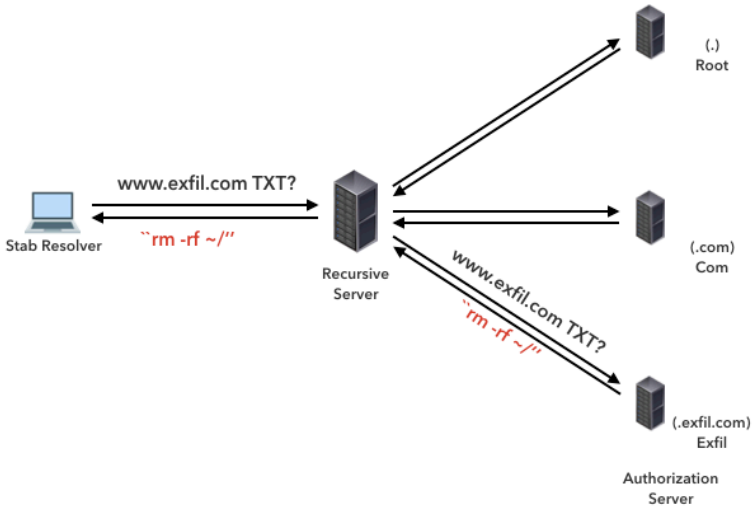

図 4 TXT レコードに登録された情報について，DNS クエリで問い合わせることで権威サーバから命令情報を取得している様子

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

本節では，DNS Tunneling に対する先行研究のアプローチを紹介する．

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，第 2.2 項で述べるような特性が出現する．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 [14] は，

3.2 ポスト名前解決システム

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

3.3 課題

本節では，先行研究および新しいアーキテクチャに基づく名前解決システムにおける DNS Tunneling への課題を示す．(大筋) 検知に基づく手法は，誤検知が避けられない．ペイロードアナリシスに対しては，一回あたりの転送量を調整することでバイパスすることができる．分析の対象がトラフィックの場合は，頻度を長期間に延長することで，ログファイルの肥大によるストレージの過去のログファイルとの分析コストを重くなり分析の隙間をバイパスできる．既存のアーキテクチャに基づくアプローチは，既存の DNS と根本から異なるアーキテクチャを採用しており，マイグレーションが考慮されていない．また，ピュアの P2P アーキテクチャおよびブロックチェーンのアーキテクチャでは，スタブリゾルバからのクエリが権威サーバとブロックチェーンをそれぞれ介することで，依然として，DNS Tunenling として機能し，発生を抑止するメソッドではない．

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

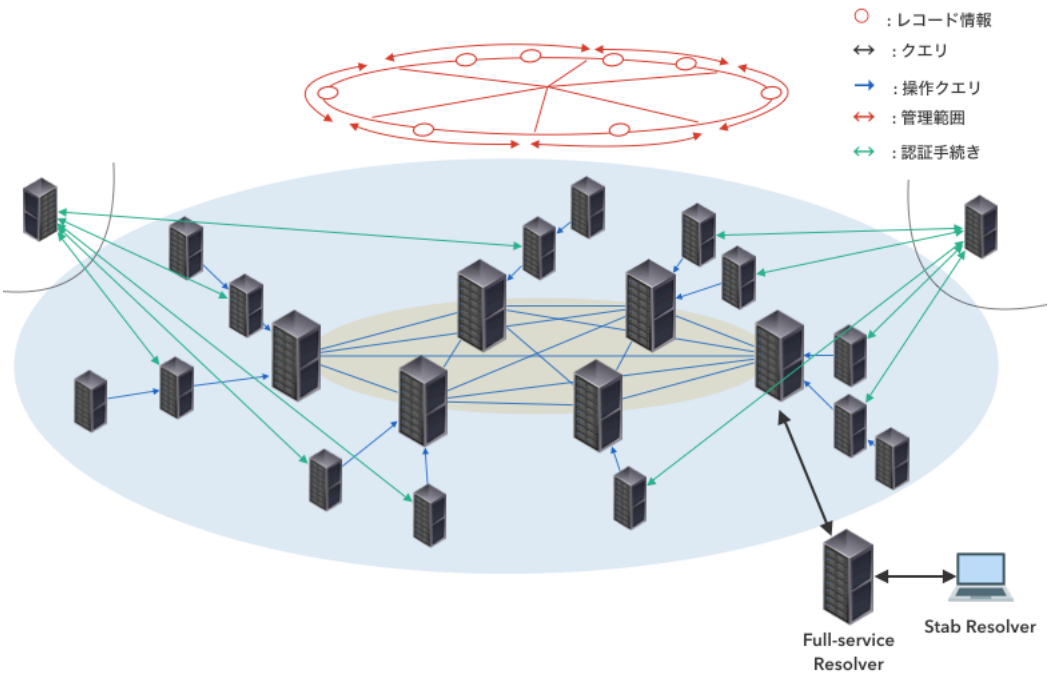

図 5 REFRES の全体図

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．

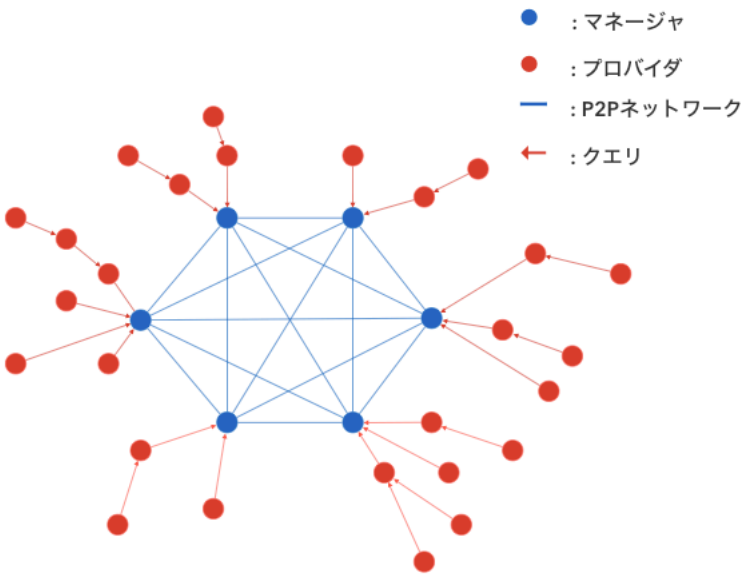

図 6　マネージャとプロバイダの関係図

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．REFRES における用語については，表 4.2.1 で示す．

表記	意味もしくは機能
コンテンツ	・識別子に関連づけられたレコード情報の実体
コンテンツ ID	・識別子
レコード情報	・リソースレコードの具体的な値 (E.g. IP アドレス)
リソースレコードタイプ	・オブジェクトに関連づけるリソースレコードの型 (E.g. A, AAAA, MX)
オブジェクト	・問い合わせる対象 (E.g. ドメイン名もしくは IP アドレス)
スタブリゾル	・名前解決クライアント
フルサービスリゾルバ	・スタブリゾルバからのクエリハンドリング ・識別子の作成
マネージャ	・フルサービスリゾルバからのクエリハンドリング ・ゾーンの管理 ・コンテンツの保持
プロバイダ	・コンテンツの作成・更新・削除操作

表 2 REFRES における用語

4.2.2 ハッシュ関数に基づくレコード情報の識別子とハッシュアルゴリズム

本項では，レコード情報にアクセスするために付与された識別子であるコンテンツ ID の説明と，その名前空間について説明する．既存の DNS の名前解決システムでは，正引きをする際，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が求まる．他方，REFRES におけるレコード情報へのアクセス方法は，識別子をキーとするこ

とでデータを取得することができる．コンテンツ ID と呼ぶ，この識別子は，ドメイン名とレコードタイプの順番でその文字列の和を引数とするメッセージダイジェストで表現することができる．例えば，ドメイン名を “www.example.com”，レコードタイプを “A” とした場合，引数となるのは “www.example.comA” という具合である．

続いて，REFRES が採用するハッシュアルゴリズムについて説明する．REFRES では，全てのコンテンツ ID がフラットな名前空間上にマップされる．従来では，異なるリソースレコードタイプのレコード情報について一つのゾーンファイルに記述することで，管理することができた．他方で，REFRES では，ドメイン名とリソースレコードタイプの組をハッシュ値の引数とするため，コンテンツ ID は，レコード情報の数に比例して増加する特性がある．また，識別子の引数の一つにドメイン名が含まれていることから，識別子から元のメッセージが導き出くことが困難な性質を備えていなくてはならない．この性質を満たすことで，なんらかの方法で識別子を悪意の第三者が取得した際に DNS Exfiltration として機能してしまうことを抑止することができる．最後に，REFRES では，既存の DNS のプロトコルフォーマットを流用する設計になっているため，識別子が格納される DNS の Question Section の QNAME のサイズ制約を満たさなくてはならない．QNAME は，最大 255bit とする任意長の領域である．以上のことを踏まえて，本システムにおけるハッシュアルゴリズムの要件は，以下の通りである．

1. 名前空間は不足を無視できる程度に大きくなくてはならない
2. アルゴリズムは一方向性の性質を備えなくてはならない
3. 最大 253bit の出力長を満たさなくてはならない

ここで，既存のハッシュ関数の特性をまとめた table を示す．

上記の内容から，REFRES では，224bit の名前空間をもつ sha3 のアルゴリズムを採用する．

続いて，分離連鎖法と 2 重ハッシュ法によるコリジョン対策方法について説明する．REFRES では，コンテンツのストアリングフェーズで ID にコリジョンが発生した場合，分離連鎖法に基づきストアされるハッシュテーブルに連結リストという形式でコンテンツがストアされる．リスト構造で延長するコンテンツの識

別には，ドメイン ID を識別子として利用する．ドメイン ID は，コンテンツ ID と同様のハッシュアルゴリズムを用いて算出されるメッセージダイジェストの先頭 32bit で表現される，ドメイン名を引数として生成される識別子である．例えば，ドメイン名を “www.example.com” とする場合，そのメッセージダイジェストが “86ff20100c058b857bae9785bf0267e6c6afb740c18b8e9a87258485” であるとすると，“86ff” がドメイン ID となる具合である．このように算出されたドメイン ID は，DNS の Question Section のうち，それぞれ 16bit 分の領域を持つタイプとクラスの領域に埋め込まれる．上記の仕組みによって，コリジョンが発生した際には，ドメイン ID をキーとしてコンテンツを識別する．

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

本項では，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 3 によって解決

される．REFRESでは，全てのサービスノードがこの対応表を保持できることを想定しており，ノードは識別子に基づきどのマネージャがコンテンツを保持しているのかを一意に特定する．

ゾーン	マネージャアドレス	ドメイン
(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 3　6 つのマネージャによって管理されるハッシュテーブルにおいて，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の例

4.2.4 認証基盤に基づく信頼されるレコード情報

本項では，レコード情報の信頼性担保のための認証基盤について説明する．

REFRESでは，全てのコンテンツについて，信頼される第三者からストアしても良いと認可されていることを前提としている．すなわち，ハッシュテーブル上のコンテンツへの操作，またはコンテンツをハッシュテーブル上にストアするなどの操作処理をする際，プロバイダは，信頼される第三者からのレコード情報に操作することを認可してもらう必要がある．認可の証明書を発行する認証局は，プロバイダの基本情報とレコード情報に基づき証明書の発行を決定する．いま，ドメイン名 “www.example.com”のリソースレコードタイプ A として “93.184.216.34” というレコード情報を関連づけるとする．プロバイダは，認証局に対して証明書発行リクエストを転送する．認証局は，リクエストされたレコード情報についてIP アドレスの到達性と不審な文字列が含まれていないこと，利用目的について評価を施す．認可された場合には，その証にディジタル証明書を発行し，マネージャにストアリクエストを転送する．マネージャは，ディジタル証明書に付与さ

れた署名に基づきコンテンツの完全性を評価し，認証された場合コンテンツ ID を計算し，担当マネージャにストアリクエストを転送する．上記の手続きを経たコンテンツがハッシュテーブル上にストアされる．

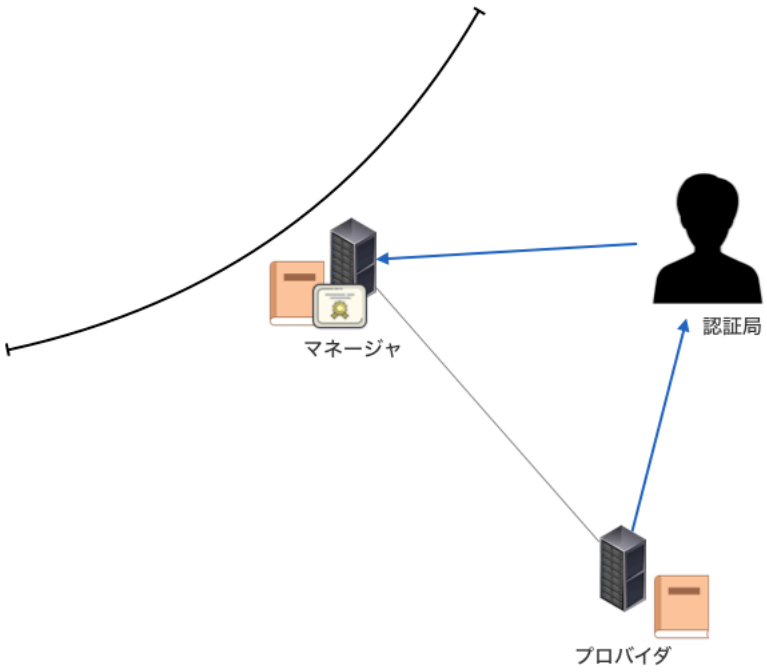

図 7　レコード情報に対する認証プロセス

4.2.5　強い型付けのレコード情報による DNS Infiltration

本項では，REFRES で使用するリソースレコードのタイプについて説明する．

はじめに，REFRES における DNSSEC の位置づけについて述べる．DNSSEC [3] は，権威サーバからの応答パケットの偽装を検知することを目的として，データの作成元の確認とデータの完全性および，不在情報応答情報の証明する DNS の拡張仕様である．これは，主として DNS の応答パケットを偽装できる程度のパラメータであることに起因する．他方で，REFRES では，応答パケットに 224bit のメッセージダイジェストが含まれるため，悪意のある応答パケットをフルサービスリゾルバに意図的にキャッシュすることは極めて困難である．以上の理由か

ら，REFRES ではDNSSEC の目的にそぐわないため，リソースレコードとして使用されない．

次に，DNSSEC 以外のリソースレコードについて説明する．第 2.2.2 項で示すように，既存の名前解決システムでドメインに関連づけることができるリソースレコードのいくつかのタイプは，DNS Infiltration として機能することができる．DNS Infiltration を抑止するリソースレコードであることの必要条件は，ドメインに関連のない任意の文字列がレコード情報に含められないことである．既存のDNS のリソースレコードのタイプのうち，任意の文字列を含めることができるのタイプは以下の通りである．

タイプ	意味	Infiltration の手法
A	ホストの IPv4 アドレス	
NS	権威サーバ	
MF	メール転送サーバ	
CNAME	別名	
SOA	権威ゾーンの開始	
NULL	NULL(実験用)	
PTR	ドメイン名のポインター (逆引き)	
MX	メール交換	
TXT	任意文字列	
DNSKEY		

表 4 DNS Infiltration として利用することができるリソースレコード一覧

上記の DNS Infiltration として機能する可能性のあるリソースレコードのタイプのうち，IP アドレスを偽装して情報を転送するものについては，第 4.2.4 項で述べた認証基盤によってレコード情報の正当性評価でスクリーニングすることができる．他方で，任意の文字列を注入できるのが，“TXT”・“NULL”・“CNAME”である．

TXT について考える．

4.2.6 コンテンツのデータフォーマット

本項では，マネージャにて管理されるコンテンツのフォーマットについて説明する．

```
{
 "ContentID" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
 "Value" : ["DomainID", "QNAME", "RType", "TTL", "Record Infomation", "Certificate Signature"]
},
{
 "contentID" : "47d87f616f550758706e1633f4de3bf28f049814eb0a52da6b554cb6",
 "message" : ["86ff", "www.example.com","A",3800,"93.184.216.34", '']
},
```

図 8 コンテンツファイル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

本項では，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

REFRESでは，224bitを固定長とするコンテンツIDをシンボルとすることによって，レコード情報にアクセスする．この仕組みの影響で，REFRESのパケットは従来のパケットと比較して肥大する特性がある．このため，送信元を目的ホストと偽装することで目的ホストの計算リソースを圧迫するDDoS攻撃に対して，脅威を高める可能性が予想される．

5.4.2 評価要素：RTT(Round Trip Time)

従来のDNSでは，ラベルごとにゾーンが移譲されている場合，レコード情報を保持するノードまでのホップ数はラベル数nに対してO(n)になる．対して，REFRESでは，識別子から一意にレコード情報を保持するマネージャノードを特定できるので，O(2)のホップ数となる．このため，既存の名前解決システムより速度の向上が期待される．

5.4.3 評価要素：トラフィック量

REFRESでは，シンボル志向の名前解決メソッドによって，既存の再帰問い合わせによるメソッドよりも少ないトラフィックに抑えることが期待される．他方

で，新たにマネージャ間通信という従来にはないトラフィックが発生する．本項では，これトラフィックがネットワーク全体にどの程度影響を及ぼすのかについて評価する．

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] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose. “DNS Security Introduction and Requirements, ” March 2005. Updated by RFCs 6014, 6840.

[4] 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).

[5] S. Bortzmeyer. “DNS Privacy Considerations RFC 7626 (INTERNET STANDARD), ” August 2015.

[6] KrebsonSecurity. “Deconstructing the 2014 Sally Beauty Breach,” May 2015. https://krebsonsecurity.com/2015/05/deconstructing-the-2014-sally-beauty-breach/. (accessd 2019-11-30).

[7] IronNet. “Chirp of the PoisonFrog,” February 2019. https://ironnet.com/blog/chirp-of-the-poisonfrog/. (accessd 2019-11-30).

[8] Nick Hoffman. “BernhardPOS,” July 2015. https://securitykitten.github.io/2015/07/14/bernhardpos.html. (accessd 2019-11-30).

[9] Fireeye. “MULTIGRAIN – Point of Sale Attackers Make an Unhealthy Addition to the Pantry,” April 2016. https://www.fieeye.com/blog/threat-research/2016/04/multigrain_pointo.html. (accessd 2019-11-30).

[10] 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/. (accessd 2019-11-30).

[11] Kaspersky. “Use of DNS Tunneling for C&C Communications,” April 2017. https://securelist.com/use-of-dns-tunneling-for-cc-communications/78203/. (accessd 2019-11-30).

[12] CISCO Talos. “Spoofed SEC Emails Distribute Evolved DNSMessenger,” October 2017. https://blog.talosintelligence.com/2017/10/dnsmessenger-sec-campaign.html. (accessd 2019-11-30).

[13] Cylance. “Threat Spotlight: Inside UDPoS Malware,” Febrary 27 2018. https://threatvector.cylance.com/en_us/home/threat-spotlight-inside-udpos-malware.html. (accessd 2019-11-30).

[14] 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.

[15] 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.

[16] 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/ICESS, Sydney, NSW, 2017, pp. 339-346.

[17] 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.

[18] 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.

[19] 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.

[20] “OzymanDNS - Tunneling SSH over DNS,” https://room362.com/post/2009/2009310ozymandns-tunneling-ssh-over-dns-html/, (accessed 2019-11-20).

[21] “iodine,” http://code.kryo.se/iodine/, (accessd 2019-11-20).

[22] “DNScat2,” 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.