Vol.2016-CSEC-74 No.26
Vol.2016-SPT-19 No.26
2016/7/15

ナイーブベイズ分類器を用いたDNS Water Torture攻撃のフィルタリング手法に関する検討

吉田 琢朗[1,a)]　竹内 優也[2]　小林 良太郎[1]　加藤 雅彦[3]　岸本 裕之[4]

概要：
本研究ではDNS権威サーバへのDDoS攻撃の一種であるDNS Water Torture攻撃による問い合わせを自動でフィルタリングするシステムの検討を行う．Water Torture攻撃は攻撃対象のドメイン名にランダムなサブドメイン名を付加した問い合わせをオープンリゾルバ状態であるキャッシュサーバに対し送信することで，多数の問い合わせを攻撃対象の権威サーバに集中させ過負荷状態とする攻撃である．本研究はナイーブベイズ分類器を用い問い合わせドメイン名のランダム性の有無より攻撃の検出を行うことを目的としており，本稿ではフィルタリングを行うシステム構成の検討と予備評価の結果について記述する．

Study on Filtering Method for the DNS Water Torture Attack Utilizing the Naive Bayes Classifier

Takuro YOSHIDA[1,a)]　Yuya TAKEUCHI[2]　Ryotaro KOBAYASHI[1]　Masahiko KATO[3]　Hiroyuki KISHIMOTO[4]

1. はじめに

Domain Name System (DNS) は，インターネットにおいてドメイン名とIPアドレスの関連付けを管理する役割を果たす極めて重要なシステムである．コンピュータやインターネットが私達の生活に深く根付いている現状を考えると，このDNSに障害が発生した場合の影響は大きいと言える．特に，大量アクセスによりDNSへの障害を引き起こすことを目的とした，Denial of Service (DoS) 攻撃やDistributed DoS (DDoS) 攻撃は脅威である．

DNSはルートサーバを頂点とする階層構造を持つ分散型システムとして構成されており，オリジナルとなるレコードは世界各地に存在する権威サーバが保持している．権威サーバは，自身が権威を持つゾーンについて，ドメイン名の情報を保持しているほか，下位ゾーンを管理する権威サーバのアドレスについても保持している．さらに，権威サーバのようにオリジナルの情報は保持しないものの，問い合わせの結果を一時記憶し，同じ問い合わせを高速に行うことを目的として設置されるキャッシュサーバが存在する．キャッシュサーバはある組織のネットワーク内部に配置され，組織内のコンピュータからの問い合わせにのみ対応する．

DNSのクエリは送信先により種類が異なり，権威サーバへは反復クエリ，キュッシュサーバへは再帰クエリが送信される．反復クエリを受け取った権威サーバは，自身の管理するゾーンに関する問い合わせであれば自身の持つ情報を返答する．下位ゾーンに関する問い合わせであれば下位の権威サーバの情報等を返却し，その権威サーバに問い合わせるよう送信元に促す．再帰クエリを受け取ったキャッシュサーバは，ドメイン名に関する最終的な回答 (正引きであればドメイン名に対応するIPアドレス) を得られるまで，上位の権威サーバから順番に反復クエリを送信し続ける．

[1] 豊橋技術科学大学 大学院 工学研究科 情報・知能工学専攻 Department of Computer Science and Engineering, Graduate School of Engineering, Toyohashi University of Technology
[2] 株式会社カーネル・ソフト・エンジニアリング Kernel Soft Engineering, Inc.
[3] 長崎県立大学 情報システム学部 情報セキュリティ学科 Department of Information Security, Faculty of Information Systems, University of Nagasaki
[4] 株式会社コムワース ComWorth Co., Ltd.
[a)] yoshida.takuro@ppl.cs.tut.ac.jp

キャッシュサーバは制限なく再帰クエリを受信するべきではない．しかしながら，一部のキャッシュサーバは，設定上の問題などからインターネット上の不特定のクエリを受け付ける状態となっているものがある．また，市販のホームルータは一般的に LAN 内のコンピュータからの問い合わせを ISP など上位ネットワークのキャッシュサーバに中継する DNS フォワーディング機能を備えている．これらホームルータの中にも，脆弱性や設定上の問題により不特定のネットワークからの問い合わせを受け付けるものが存在する．このように不特定の問い合わせを受け付けるキャッシュサーバやホームルータを，オープンリゾルバと呼ぶ．

このオープンリゾルバを踏み台とする DNS への DDoS 攻撃の一つに，Water Torture 攻撃がある．キャッシュサーバはキャッシュしていないドメイン名の問い合わせを要求されると，必ず権威サーバに問い合わせを行うこととなる．この挙動を悪用し，ドメイン名として過去に問い合わせをされていないランダムな文字列を含むクエリをオープンリゾルバに送信し，踏み台とすることで，権威サーバに大きな負荷をかける．この攻撃手法は 2014 年頃発見されたものであり，攻撃への対策は急務となっている．

Water Torture 攻撃発生時，権威サーバには大量のクエリが送信されるため，その対策手法には精度のみならず高速性も求められる．そこで本研究では，クエリを解析し問い合わせドメイン名にランダムな文字列が含まれるかを高速に判定することで，Water Torture 攻撃に関係のあるクエリを識別する手法を提案する．また，その手法を応用し，権威サーバの前段に配置して攻撃クエリを自動的にフィルタリングするシステムについて検討する．

ランダムなクエリの判定処理においては，将来的に FPGA や GPGPU などハードウェアでの実装を想定し，アルゴリズムが単純で大量処理に向くナイーブベイズ分類器を採用する [1]，[2]．

本稿では，提案する検出手法の検出率について実験を行い，評価する．

本稿の構成を示す．第 2 章で DNS に関係する攻撃を取り上げた関連研究，および Water Torture 攻撃の概要について述べる．第 3 章で本研究において提案する攻撃の判定手法について説明し，第 4 章でその実装の詳細を記述する．第 5 章で実験により得られた提案手法の検出率を提示し，それに対し第 6 章で考察を述べる．第 7 章で今後の展望を述べ，第 8 章でまとめを行う．

2. 関連研究

2.1 DNS における悪意のある挙動の検出

Okayasu らは Command-and-Control (C&C) サーバのドメイン名が逆引き可能か否か， Time to Live (TTL) 値の大きさ，および，アドレスレコード (A レコード) の内容を検証することで，ボットネットによる通信を検出する方法を提案している [3]．この手法は不自然なドメイン情報に注目しているが，全ての DNS への攻撃がこのような特徴を持つわけではない．Water Torture 攻撃においては，登録されているドメインの情報ではなく，存在しないドメイン名を問い合わせるクエリの頻度が問題となる．そこで，我々の研究ではドメインの情報は考慮せず，サーバが受け取るクエリのドメイン名を解析することで Water Torture 攻撃を検出する．

Karasaridis らは DNS キャッシュポイズニングや DNS トンネル攻撃の検知方法を提案している [4]．彼らの研究は，侵入検知システムと異なり，シグネチャなどの既知の攻撃パターンに依存せず，IP アドレス，送信元・受信元のポート番号，パケット数，バイト数などの情報を含むフローデータを利用している．一方で，これらの情報だけでは，Water Torture 攻撃の判定は難しい．そのため，我々の方法では上記の情報ではなく，クエリのドメイン名を解析の対象とした．

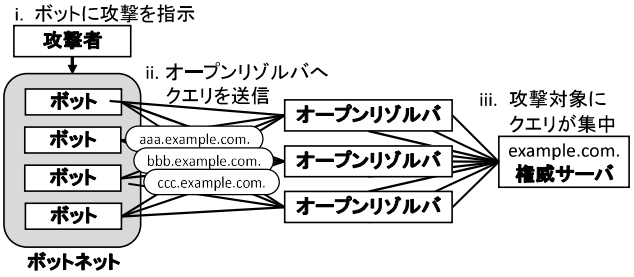

図 1　Water Torture 攻撃クエリの流れ

2.2 Water Torture 攻撃

Water Torture 攻撃 (水責め攻撃，ランダムサブドメイン攻撃) は，DNS サーバへの DDoS 攻撃の一種である．この攻撃が発生した場合，権威サーバとキャッシュサーバのどちらにも被害を及ぼす可能性があるが，一般的に，攻撃対象は特定の権威サーバであるとされている [5]．

図 1 は攻撃クエリの流れを示している．攻撃者は，多数のボットを用意し，ボットからオープンリゾルバに何度も攻撃対象のドメイン名に関する問い合わせを行う．この時，送信されるドメイン名には，サブドメインとしてランダムな文字列が設定されているという特徴を持つ．クエリを受け取ったオープンリゾルバは，送信されたドメイン名がキャッシュに存在しないため，必ず権威サーバに問い合わせることになる．多数のオープンリゾルバに向けて送信されたクエリは，最終的に攻撃対象の権威サーバに集中し，そのサーバを過負荷状態にさせ，サービス停止に追い込む．

権威サーバ側で行うことのできる一般的な Water Torture 攻撃への対策としては，大量にクエリを送信するキャッシュサーバからの問い合わせを IP アドレスにより識別し

て拒否する対策や，一定時間内に受け付けるクエリの数を制限するレート制限などが考えられる．しかし，これらは Water Torture 攻撃と関係のない一般ユーザをも巻き込む可能性がある．

前述のオープンリゾルバ化したホームルータが攻撃の踏み台とされている場合，権威サーバは，Water Torture 攻撃によるクエリが ISP の所有するキャッシュサーバから大量に送信されていると認識する．そのため，それらのキャッシュサーバからの問い合わせを拒否すると，その ISP と契約している一般ユーザも名前解決が不能となる．

また，レート制限を行った場合では，攻撃が続く限りは一般のユーザが拒否される可能性がある．

つまり，これら一般的な手法は Water Torture 攻撃の根本的な解決にはならず，その対策には Water Torture 攻撃により発生したクエリを識別し選択的に排除することが必須である．

Water Torture 攻撃によるクエリを検出するために，我々は問い合わせドメイン名に含まれるランダムな文字列の存在に着目した．ランダムなドメイン名の検出に関する研究は，既に存在している．

Kazato らは，ドメインが存在しないことを示すエラー (NXDOMAIN) を示すレスポンスを分類し，その発生原因について考察している [6]．彼らの研究では，NXDOMAIN であるレスポンスから問い合わせドメイン名を抽出し，それらを 9 パターンに分類している．そのパターンの一つは，ドメイン名にランダムな文字列を含むものである．ランダムであるかの判定には，クエリのドメイン名と正しいドメイン名を比較することで算出したバイグラムベースのスコアを用いている．この研究はあくまでも存在しないドメイン名に関して調査したものであり，特定の攻撃を判定することを目的としたものではない．我々はこのスコアリングは Water Torture 攻撃の判定にも適用可能であると考えた．そこで，本研究においても，ランダム性を判定する際の特徴量の一つとして，バイグラムベースのスコアを用いることとした．

Yadav らは，ボットネットの検出のため，ボットと C&C サーバの通信時に使用されるドメイン名に注目した研究を行っている．C&C サーバが使用するドメイン名は，Domain Generation Algorithm (DGA) によって生成される．そして，一般的なサーバで使用されるドメイン名と異なり，短時間でドメイン名が変わることが多い．そこで，彼らはドメイン名に使用された文字種の確率分布から Kullback-Leibler (K-L) 情報量を求め，それにより DGA によって生成されたドメイン名を検出している．対して我々の研究は，ボットネットの検出ではなく Water Torture 攻撃の検出を行う点，検出率だけでなく検出速度についても考慮している点に違いがある．

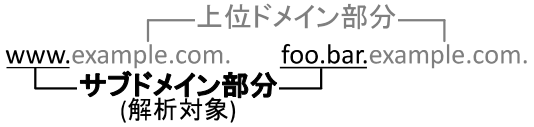

図 2　解析対象

3. 提案手法

3.1 判定アルゴリズム

本研究において，解析処理の対象となるのはクエリの完全修飾ドメイン名 (FQDN) のみである．また，Water Torture 攻撃であるか判断する基準は FQDN のうちサブドメイン部分にランダム文字列を含むか否かであるため，上位ドメイン部分は解析対象としないこととする (図 **2**)．

サブドメインのランダム性を判定するために，ナイーブベイズ分類器を用いる．ナイーブベイズ分類器の詳細は 3.3 節で述べる．ナイーブベイズ分類器は，特徴量同士が独立と仮定することで，線形時間での解析が可能である [7]．

我々は高速なトラフィックをリアルタイムにフィルタリングすることを重視しているため，解析処理を FPGA や GPGPU などで高速に処理する手法が確立していることが望ましい．ハードウェアでの処理が可能な分類アルゴリズムには，他に決定木学習などが存在するが [8]，本研究においては，実装の容易性という観点からナイーブベイズ分類器を選択した．

ナイーブベイズ分類器は教師有り学習アルゴリズムである．本手法において，訓練データはインプットとそれに対応するアウトプットの組である．インプットは各特徴量の値，アウトプットはランダムな文字列を含むか否かである．学習により，ランダム用参照テーブルと非ランダム用参照テーブルを作成する．参照テーブルとは，解析処理時に参照される特徴量の度数分布表であり，絶対度数 (その階級の範囲内の特徴量の出現数) および相対度数 (確率) の情報を含む．参照テーブルの各項目の階級は 3.2.1 節に示す．

学習後，システムは解析対象データを逐次読み取り，その特徴量を元にランダム用および非ランダム用参照テーブルを参照し，ランダムである確率，および非ランダムである確率を算出し，その大小でランダムか否かを判定をする．

3.2 特徴量

本研究では，以下の要素を特徴量として抽出する．

(1) 文字数
(2) ラベル数
(3) バイグラムベースのスコア

Water Torture 攻撃で発生するクエリは，ランダム性を確保するため，通常のドメイン名と比較し長くなると考えられる．そのため，文字数とラベル数は特徴量として適切で

表 1　階級の設定

1	長さ	1	2	3	…	9	10-
2	ラベル数	1	2-3	4-5	6-7	8-	

あると判断した．

バイグラムは文字列を 2 文字ずつ区切ったシーケンスであり，例えば，“foo.bar.” であれば，バイグラムは “fo”, “oo”, “o.”, “.b”, “ba”, “ar”, “r.” となる．そして，バイグラムベースのスコアとは，バイグラム単位で参照テーブルを参照し，その値を全て加算したものである．DNS は仕様上ドメイン名の大文字小文字を区別しない [9]．そのため，このスコアの計算に際しても大文字小文字は区別しないこととする．

3.2.1　参照テーブルの階級の設定

参照テーブルの階級の設定は，検出率を左右する重要な要素である．そこで，評価を行い最も良い検出率を実現できる階級の設定を求めた．その評価の詳細については，5.3 節で示すこととし，ここではその最適な階級の設定を**表 1** に示す．

3.3　ナイーブベイズ分類器

3.2 節で示す 3 種の特徴量を 3 次元のベクトル $\boldsymbol{X} = (x_1, x_2, x_3)$ で表現し，それをランダム，非ランダムを示す 2 つのクラス y_0, y_1 のうち最も可能性の高いクラスに分類する．これは，ランダム用参照テーブルからランダムである確率 $P(y_0|\boldsymbol{X})$，非ランダム用参照テーブルから非ランダムである確率 $P(y_1|\boldsymbol{X})$ をそれぞれ計算し，比較することで行う．

$P(y_0|\boldsymbol{X})$ を計算する場合の詳細な手順を以下に示す．

(1) 以下に示す値を，ランダム用参照テーブルより取得する:

$$P(y_0), P(x_1|y_0), P(x_2|y_0), P(x_3|y_0)$$

(2) 特徴量の全てが独立と仮定し，以下の計算を行う:

$$P(\boldsymbol{X}|y_0) = P(x_1|y_0)P(x_2|y_0)P(x_3|y_0)$$

(3) ベイズの定理より，以下の計算を行う:

$$P(y_0|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|y_0)P(y_0)}{P(\boldsymbol{X})} \propto P(\boldsymbol{X}|y_0)P(y_0)$$

同様に，$P(y_1|\boldsymbol{X})$ を計算し，$P(y_0|\boldsymbol{X})$ と比較することで，より確率の高いクラスに分類する．

4.　実装

本節では，提案手法を用いて解析対象のランダムサブドメイン名の有無を識別するプログラムの実装について述べる．

4.1　概要

プログラムは以下 2 つのフェーズを持つ．

表 2　非ランダム用参照テーブルの例 (解析対象を 1 つ読み込み後)

ラベル数		長さ		バイグラム	
1	1	1	0	“fo”	1
2-3	0	2	0	“oo”	1
4-5	0	3	1	“o.”	1
6-7	0	…	0	…	
8-	0	10-	0	計	3
非ランダム訓練データの総数: 1					

表 3　非ランダム用参照テーブルの例 [*1] (学習フェーズ終了後)

ラベル数		長さ		バイグラム	
1	1,750	1	50	“fo”	180
	(35%)		(1%)		(0.18%)
2-3	1,600	2	150	“oo”	195
	(32%)		(3%)		(0.20%)
4-5	1,000	3	250	“o.”	890
	(20%)		(5%)		(0.89%)
6-7	600	…		…	
	(12%)				
8-	50	10-	1,600		
	(1%)		(32%)	計	100,000
非ランダム訓練データの総数: 5,000 (50%)					

学習フェーズ　訓練データセットから参照テーブルを生成する．

解析フェーズ　解析対象のランダム性を判定する．

以下，具体的な例を用いて各フェーズの説明を行う．

4.2　学習フェーズ

まず，非ランダム用参照テーブルを初期化する．次に，“foo.” というサブドメイン名から抽出された特徴量 (ラベル数=1, 長さ=4, バイグラム=“fo”, “oo”, “o.”) が 1 つ与えられると，**表 2** の非ランダム用参照テーブルを生成する．それ以降，訓練データファイルの全てを読み込むまで非ランダム用参照テーブルの更新を行う．最終的に，5000 個の非ランダムデータを読み込んだ結果，**表 3** の非ランダム用参照テーブルを得る．同様に，ランダム用参照テーブルを生成する．

4.3　解析フェーズ

“foo.” というサブドメイン名に対し，表 3 を用いて非ランダムである確率を求める．

$$\begin{aligned} P(Y|\boldsymbol{X}) &\propto P(X_1|y_0)P(X_2|y_0)P(X_3|y_0)P(y_0) \\ &= 35\% \cdot 5\% \cdot (0.18\% + 0.2\% + 0.89\%) \cdot 50\% \\ &= 0.011\% \end{aligned}$$

同様に，ランダム用参照テーブルを用いてランダムである確率を求める．そして，ランダムである確率と非ランダ

[*1] 下線部の項目は 4.3 節で使用する

ムである確率を比較し，前者が高ければランダム，後者が高ければ非ランダムであると分類する．

なお，各々の確率は計算の一部を省略しているため，ランダムである確率と非ランダムである確率を合計しても100%とならない．そのため，必ず両方について確率を計算する必要がある．

これを，全ての解析対象に対して行う．

5. 評価実験

本研究で提案する手法の検出率を，実験によって確認した．

5.1 評価環境

全ての解析対象について，手動で正解データを作成し，本手法での解析結果と比較することで，検出率 (正解率) と偽陽性・偽陰性の発生率を評価した．

5.2 データセット

評価に使用した訓練データセットおよび解析対象データセットは，以下をそれぞれ同数含んだものである．

キャプチャデータ　パケットキャプチャを行い取得したクエリから作成 *2

生成データ　プログラムにより生成したランダムなドメイン名から作成

訓練データセットで使用したキャプチャデータは，以下の期間にキャプチャされたものを使用した．

DS-0　2016 年 3 月 2 日 10 時 25 分 40 秒
～ 3 月 3 日 10 時 25 分 39 秒

解析対象データセットは 3 種用意し，それぞれ以下の期間にキャプチャした．

DS-1　2016 年 3 月 3 日 10 時 25 分 40 秒
～ 3 月 4 日 10 時 25 分 39 秒

DS-2　2016 年 3 月 4 日 10 時 25 分 40 秒
～ 3 月 5 日 10 時 25 分 39 秒

DS-3　2016 年 3 月 5 日 10 時 25 分 40 秒
～ 3 月 6 日 10 時 25 分 39 秒

RFC1035[9] に従い，生成データのドメイン名は次の特徴を持つ．

- 英数字で構成する
- 1 つ以上のラベルを持つ
- 各ラベルは 1 文字以上 64 文字以下である
- 文字数が 255 文字以下である

5.3 特徴量の選択

各特徴量の有効性を確認するため，使用する特徴量の組合せを変化させた場合の検出率を比較した．**表 4** に示すように，提案する 3 種の特徴量全てを使用した場合，最も検出率が高い．そのため，全ての特徴量が有効に作用していると考える．

なお，具体的な結果の記載は省略するが，各特徴量の参照テーブルでの階級についても，表 1 と異なる設定で検出率を比較した．そして，表 1 が最も良い結果が得られることを確認した．

*2　豊橋技術科学大学のエッジルータを通る DNS クエリを全てキャプチャした

表 4　特徴量の選択と正解率・偽陰性率・偽陽性率

特徴量の組合せ *3	正解率 (%)	偽陽性率 (%)	偽陰性率 (%)
1,2,3	96.85	2.79	0.36
1,2	85.40	4.73	9.87
1,3	73.59	0.08	25.61
2,3	95.69	3.74	0.58
1	81.23	1.05	17.72
2	95.99	3.84	0.18
3	95.73	3.74	0.53

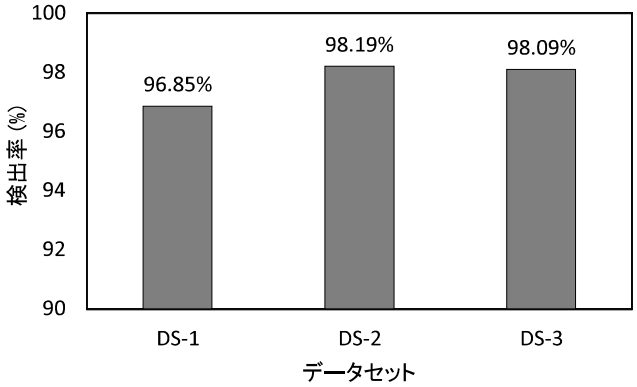

図 3　データセット毎の正解率

表 5　DS-1 での評価結果の混同行列 *4

		解析結果	
		ランダム	非ランダム
正解	ランダム	50.03% (98,675)	0.36% (710)
	非ランダム	2.79% (5,502)	46.82% (92,348)
		正解率	96.85%
		不正解率	3.15%

5.4 結果

図 3 に各データセットでの正解率，**表 5** に DS-1 での混同行列を示す．正解率は平均 97.17%，最低でも DS-1 の場合の 96.85%であり，データセットによらず安定した正解率を実現できた．また，表 5 において，偽陰性より偽陽性が多く現れているが，これは他のデータセットにおいても同様の傾向が見られた．

*3　特徴量の番号は 3.2 節で列挙した順と一致する
*4　括弧内の数値は処理した解析対象の数を表す

6. 考察

偽陽性は，サブドメイン部分の文字数が多いものについて特に発生しやすい．一般的な通信に現れる正常なドメイン名のサブドメイン部分は，“www” など文字数の少ないものが多く，ゆえに文字数が少ないほど，非ランダムである確率が高く判定される傾向にある．

本研究では，本手法をリアムタイムのフィルタリングシステムに応用することを考えており，偽陽性率は低いことが望ましい．そこで，簡易的な対策としては，ナイーブベイズ分類器により算出されたランダムである確率，非ランダムである確率にバイアスをかけ，非ランダムであると判定されやすくさせることが考えられる．しかしこれは偽陰性を高めると考えられるため，バイアスの程度は運用者のポリシーにより変更できることが望ましい．

また，本手法では，参照テーブルと解析対象が与えられると，結果が一意に定まる．そして，学習フェーズ後は参照テーブルに対し変更が加えられることはない．つまり，本手法の結果を基に自動的にクエリのフィルタリングを行うとすると，繰り返し現れるドメイン名がランダムと判定された場合は，そのドメイン名が拒否され続けることとなる．これに対する対策としては，非ランダムと判定された問い合わせであっても，一定回数以上連続した場合は，ホワイトリストに登録し，フィルタリングを抑制するといった対策が考えられる．

ただ，上記のいずれの対策も偽陽性を根本的に下げるものではないため，検出率の向上にはさらなる特徴量の吟味が必要と考えられる．

7. 今後の展望

7.1 フィルタリングシステムの構成

本研究で提案する Water Torture 攻撃の検出手法は，図 **4** に示す構成のフィルタリングシステムに応用されることを想定している．システムは権威サーバの前段に設置し，サーバへ向かうクエリは全て本システムを経由させ，解析およびフィルタリングを行う．システムにより，クエリがランダムなサブドメイン名を含むと判断した場合，そのクエリは権威サーバに中継せず，システムが権威サーバに代わり，サーバエラー (SERVFAIL) を示すレスポンスを返却する．

7.2 解析処理での FPGA・GPGPU の活用

現時点では，全ての処理はソフトウェアで実現している．

しかし，本研究においてランダム性の判定にするナイーブベイズ分類器は，FPGA や GPGPU を用いて高速に動作させることが可能である [1]，[2]．そこで，上述のフィルタリングシステムをより高トラフィックな環境でも使用可能なものとするため，解析処理をそれらにオフロードするよう変更したいと考える．

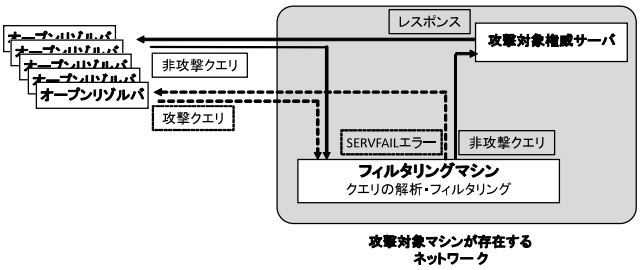

図 **4** システム構成図

7.3 キャッシュサーバへの応用

本手法では権威サーバ側でのフィルタリングを目的としていたが，同様の判定手法でキャッシュサーバへの適用も可能と考える．

本手法において，解析の対象となるクエリのドメイン名は，保護対象の権威サーバが権威を持つものに限られる．そのため，問い合わせのドメイン名の上位から権威を持つドメイン名を消去することで，容易にサブドメイン部分を取得することができる．

権威サーバとは異なり，キャッシュサーバの場合はあらゆるドメイン名に関する問い合わせを受けるため，上位ドメイン部分は自明ではない．従って，正確にサブドメイン部分を取得するためには，クリエを一度通過させ，レスポンスの権威セクションを参照する，あるいは，簡易的に下位のいくつかのラベルをサブドメイン部分として扱うなどの方法が考えられる．

8. まとめ

本稿において，我々は DNS に対する DDoS 攻撃の一種である Water Torture 攻撃の検出手法について述べた．本手法は，Water Torture 攻撃のクエリの特徴である，サブドメイン名にランダムな文字列を含むという点に着目し，ナイーブベイズ分類器を用いて，攻撃クエリの判定を行う．評価の結果，平均 97.17%の正解率で検出を行うことができると分かった．

本手法を応用し攻撃クエリをフィルタリングするシステムの実装および，パフォーマンスの測定については，今後の課題としたい．

参考文献

[1] Andrade, G., Viegas, F., Ramos, G. S., Almeida, J., Rocha, L., Goncalves, M. and Ferreira, R.: GPU-NB: A Fast CUDA-Based Implementation of Naive Bayes, *2013 25th International Symposium on Computer Architecture and High Performance Computing*, pp. 168–175 (online), DOI: 10.1109/SBAC-PAD.2013.16 (2013).
[2] Meng, H., Appiah, K., Hunter, A. and Dickinson, P.: FPGA

implementation of Naive Bayes classifier for visual object recognition, *CVPR 2011 WORKSHOPS*, pp. 123–128 (online), DOI: 10.1109/CVPRW.2011.5981831 (2011).

[3] Okayasu, S. and Sasaki, R.: Evaluation of Method for Detecting C&C Server of Botnet Using the Latest Data (in Japanese), *Proceedings of the 2014 Computer Security Symposium*, Vol. 2014, No. 2, pp. 175–182 (2014).

[4] Karasaridis, A., Meier, H. K. and Hoeflin, D.: NIS04-2: Detection of DNS Anomalies using Flow Data Analysis, *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pp. 1–6 (2006).

[5] Secure64SoftwareCorporation: Water Torture: A Slow Drip DNS DDoS Attack, , available from ⟨https://blog.secure64.com/?p=377⟩ (accessed 2015-11-30).

[6] Kazato, Y., Fukuda, K. and Sugawara, T.: Towards Classification of DNS Erroneous Queries, *Proceedings of the 9th Asian Internet Engineering Conference*, pp. 25–32 (2013).

[7] Amor, N. B., Benferhat, S. and Elouedi, Z.: Naive Bayes vs Decision Trees in Intrusion Detection Systems, *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, New York, NY, USA, ACM, pp. 420–424 (online), DOI: 10.1145/967900.967989 (2004).

[8] Kulaga, R. and Gorgon, M.: FPGA Implementation of Decision Trees and Tree Ensembles for Character Recognition in Vivado Hls, *Image Processing & Communications*, pp. 71–82 (2015).

[9] Mockapetris, P.: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, , available from ⟨ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt⟩ (accessed 2015-11-30).