～DNSのセキュリティと関連技術～

はじめに

説明にあたり、使用OSはCentOS5.4，使用DNSソフトウェアはBIND9.6を前提としています。

目次

DNSのセキュリティの基本

DNSソフトウェア

DNSソフトウェアは、なるべくバージョンが高くて安定性のあるもの(脆弱性のないもの)にする。

DNSの構成

DNSサーバは、コンテンツサーバとキャッシュサーバに分かれる。
役割が異なるため、できるなら物理的にコンテンツサーバとキャッシュサーバは分離すべき。

① コンテンツサーバ

コンテンツサーバとは、ドメインのゾーン情報を持っているDNSサーバのこと。
マスターサーバ(プライマリサーバ)とスレーブサーバ(セカンダリサーバ)に分類される。
マスターサーバは、ドメインのゾーン情報のオリジナルを持っているDNSサーバ。
スレーブサーバは、マスターサーバからゾーン情報のコピーを取得して保持しているDNSサーバ。

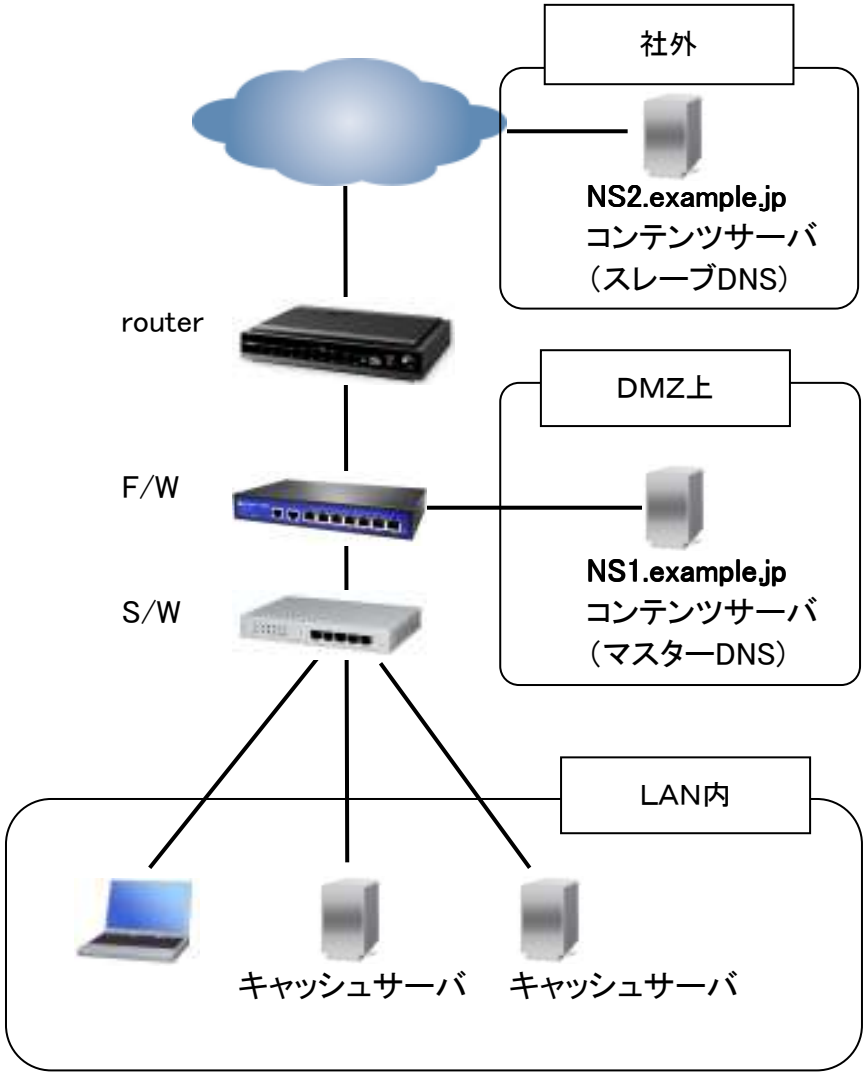
- 誰でも参照可能にする
- 外部に公開する
- 再帰問い合わせを禁止する

② キャッシュサーバ

キャッシュサーバとは、外部のDNSサーバに問い合わせをした結果をキャッシュ(保存)するDNSサーバであり、
ドメインのゾーン情報を管理していない。
そのため、基本的にユーザの問い合わせ先のDNSとして利用される。

- 利用者を制限する
- 外部に公開しない
- 再帰問い合わせを許可する

基本構成 その1



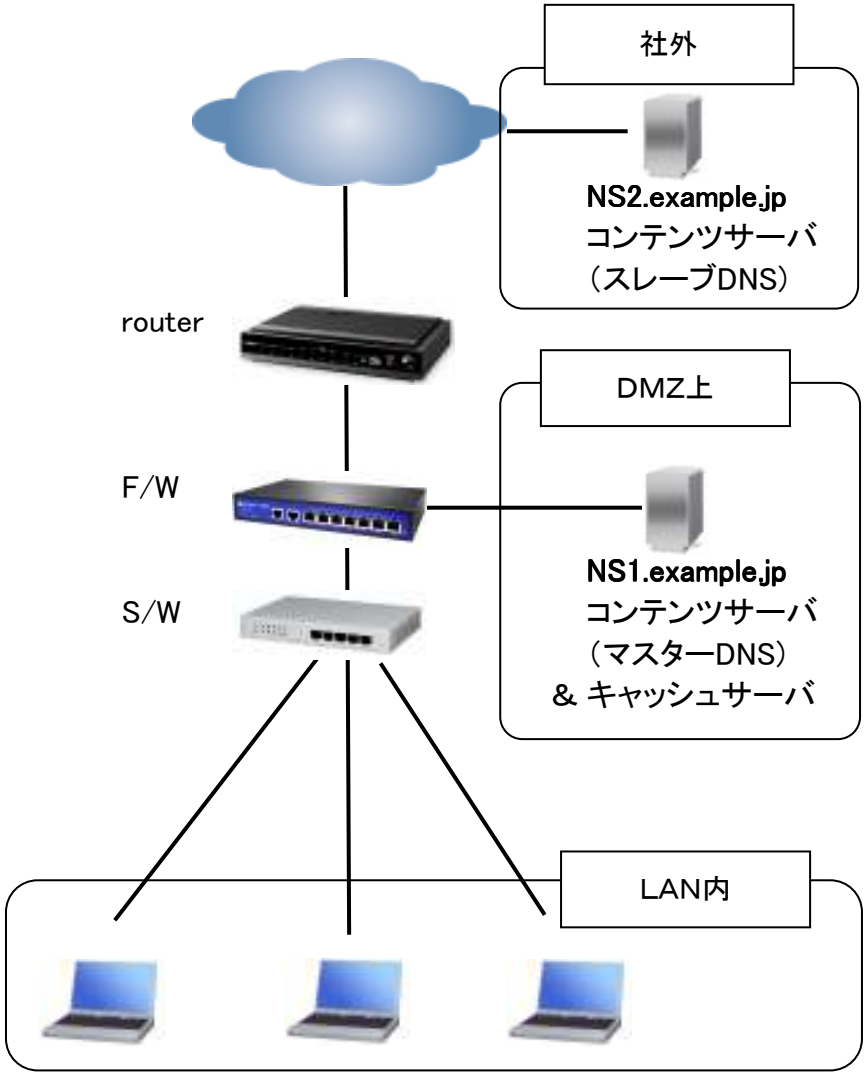
DNSの基本的な構成

「止めない」ネットワークを構成するためには、
4台以上の物理的に異なるDNSが望ましい

コンテンツサーバ　× 2台
　外部からアクセスできる場所に設置する。
　マスターサーバとスレーブサーバは、
　ネットワーク的に分離させる。
　・ マスターサーバ ・・・・・ DMZ上
　・ スレーブサーバ ・・・・・ 外部ネットワーク上
　　（委託）

キャッシュサーバ×2台
　外部からアクセスできないLAN上に設置する。
　キャッシュサーバもネットワーク的に分離させる。

基本構成 その2



社内にDNSサーバが1台しかない場合

１台でコンテンツサーバ＆キャッシュサーバ兼用

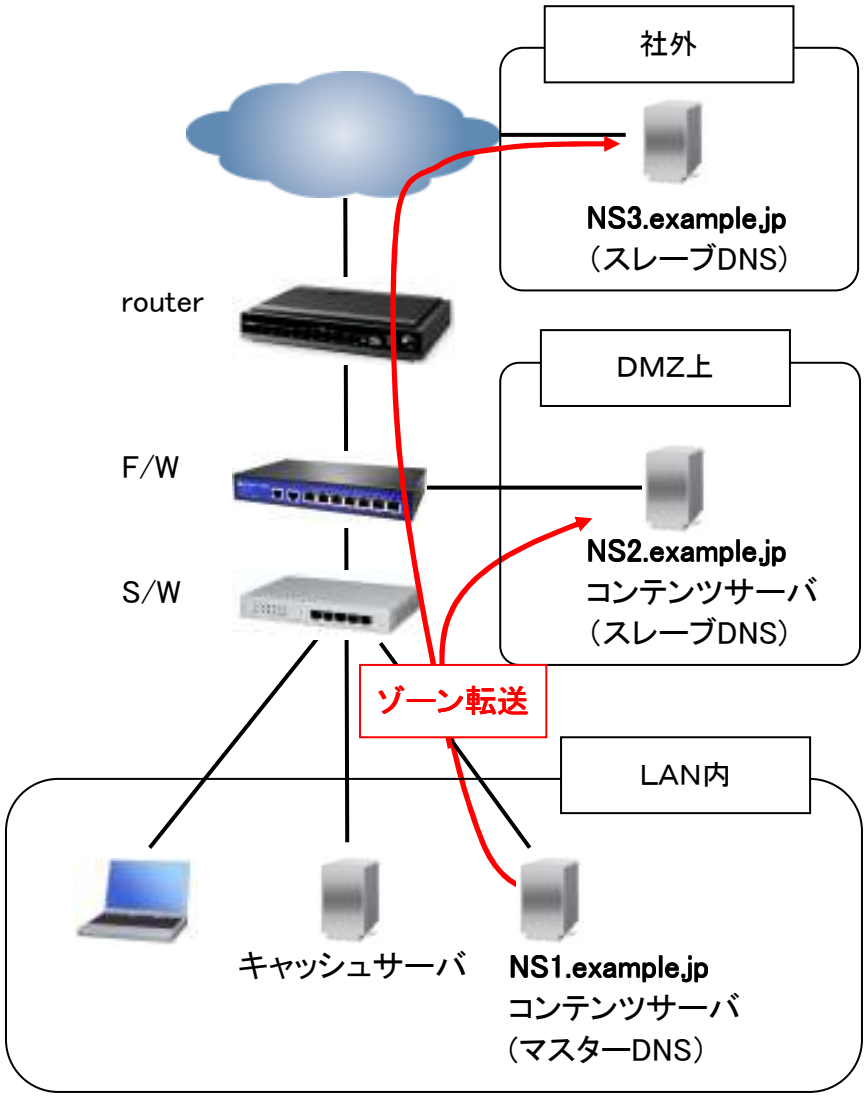
viewステートメントを使って、
外部向けには、コンテンツサーバとして公開し、
内部向けには、キャッシュサーバとして公開
サーバは、DMZ上に配置する。

コンテンツサーバのセカンダリは外部に委託。

```
view "外部向け" {
        ゾーン情報　;
        再起問い合わせ不可　;
} ;

view "内部向け" {
        問い合わせ制限　;
        再起問い合わせ許可　;
} ;
```

ヒドゥンプライマリDNS



ヒドゥンプライマリDNS

コンテンツサーバのマスターサーバを隠し(非公開)、
外部からの侵入や攻撃を防ぐこと。
具体的には、マスターサーバをNSレコードの登録から外すことにより非公開にすることができる。
マスターサーバは、外部からアクセスできないLAN上に配置して、そこから外部のスレーブサーバにゾーン情報を転送する。

example.jpゾーンファイル
[通常の設定]

```
example.jp.  IN  NS  NS1.example.jp
example.jp.  IN  NS  NS2.example.jp
example.jp.  IN  NS  NS3.example.jp
```

⇩

[ヒドゥンプライマリDNSの設定]

```
example.jp.  IN  NS  NS2.example.jp
example.jp.  IN  NS  NS3.example.jp
```

NSレコードに登録しないことにより、外部にその存在を隠す

ゾーン転送を制限する

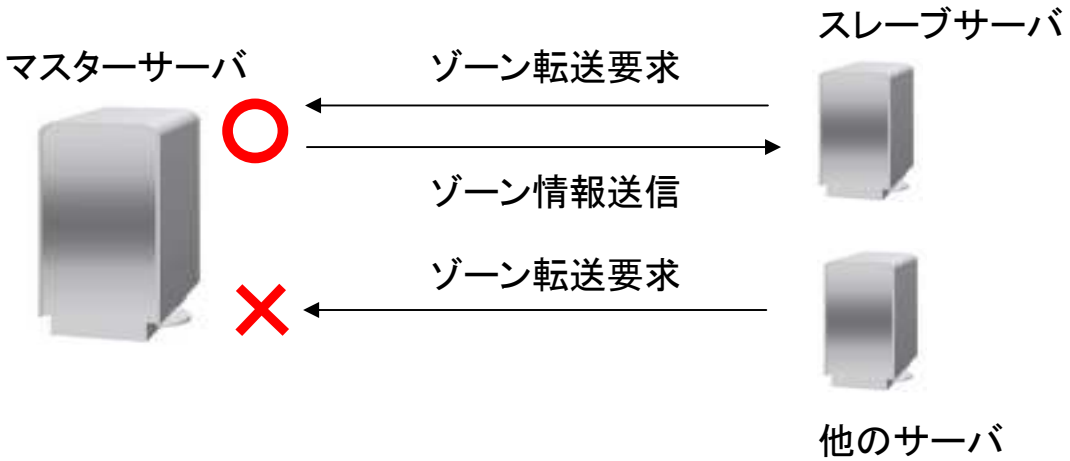
ゾーン転送とは、スレーブサーバがマスターサーバからゾーン情報を取得する方法のひとつ。
マスターサーバ側は、特定のスレーブサーバのゾーン転送要求だけに応答するように、allow-transferオプションを使って制限をかける。

マスターサーバのnamed.conf

```
#  全体で制限をする場合
   options {
           allow-transfer {
           スレーブ・サーバのIP;
           };
   };

#  zone単位で制限をする場合
   zone "ゾーン名" {
           type master;
           file "ファイル名";
           allow-transfer {
             スレーブ・サーバのIP;
           };
   };
```

※ 先頭の#は、コメント行を示す



問い合わせを許可するユーザを制限する

キャッシュサーバについては、利用するユーザを制限する。
コンテンツサーバについては、自身が管理するゾーン情報への問い合わせの制限をしてはいけない。

コンテンツサーバのnamed.conf

```
設定なし
```

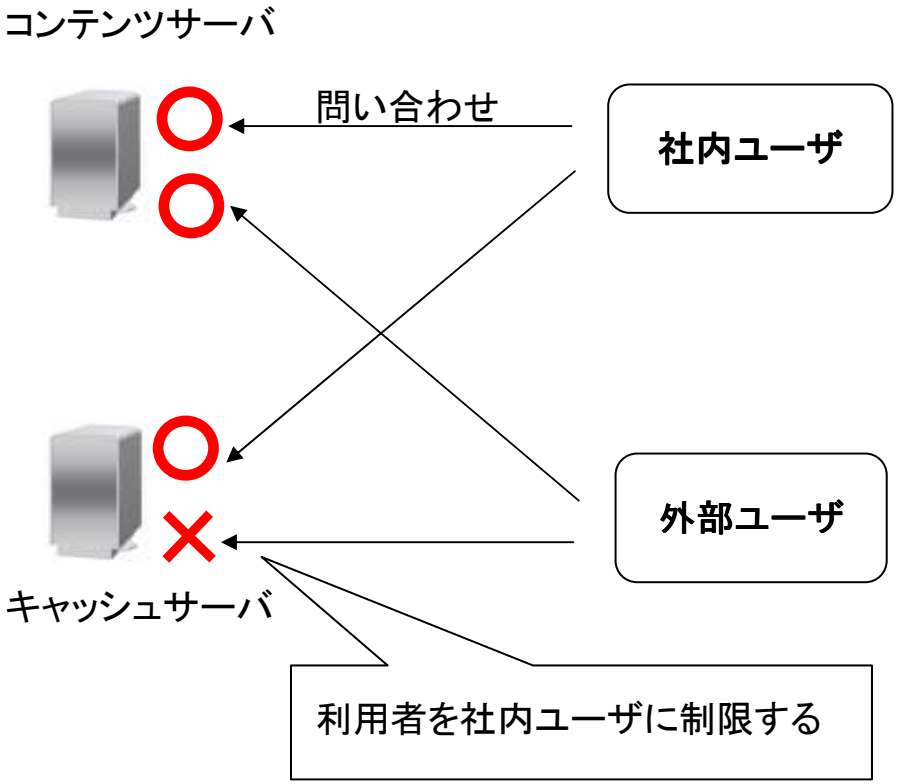
キャッシュサーバのnamed.conf

```
options {
#  問い合わせを許可するユーザ
   allow-query { IPアドレス;
                 IPアドレス/Prefix長;
                 ACL名;

    };

#  問い合わせを拒否するユーザ
   blackhole { IPアドレス;
               IPアドレス/Prefix長;
               ACL名;
    };
};
```

※ 先頭の#は、コメント行を示す



再帰問い合わせを禁止する

再帰問い合わせとは、DNSサーバが問い合わせに対する回答を知らない場合に、問い合わせ元に代わって外部のDNSサーバに問い合わせること。

キャッシュサーバについては、利用するユーザを制限する。
コンテンツサーバについては、再帰問い合わせを禁止する。

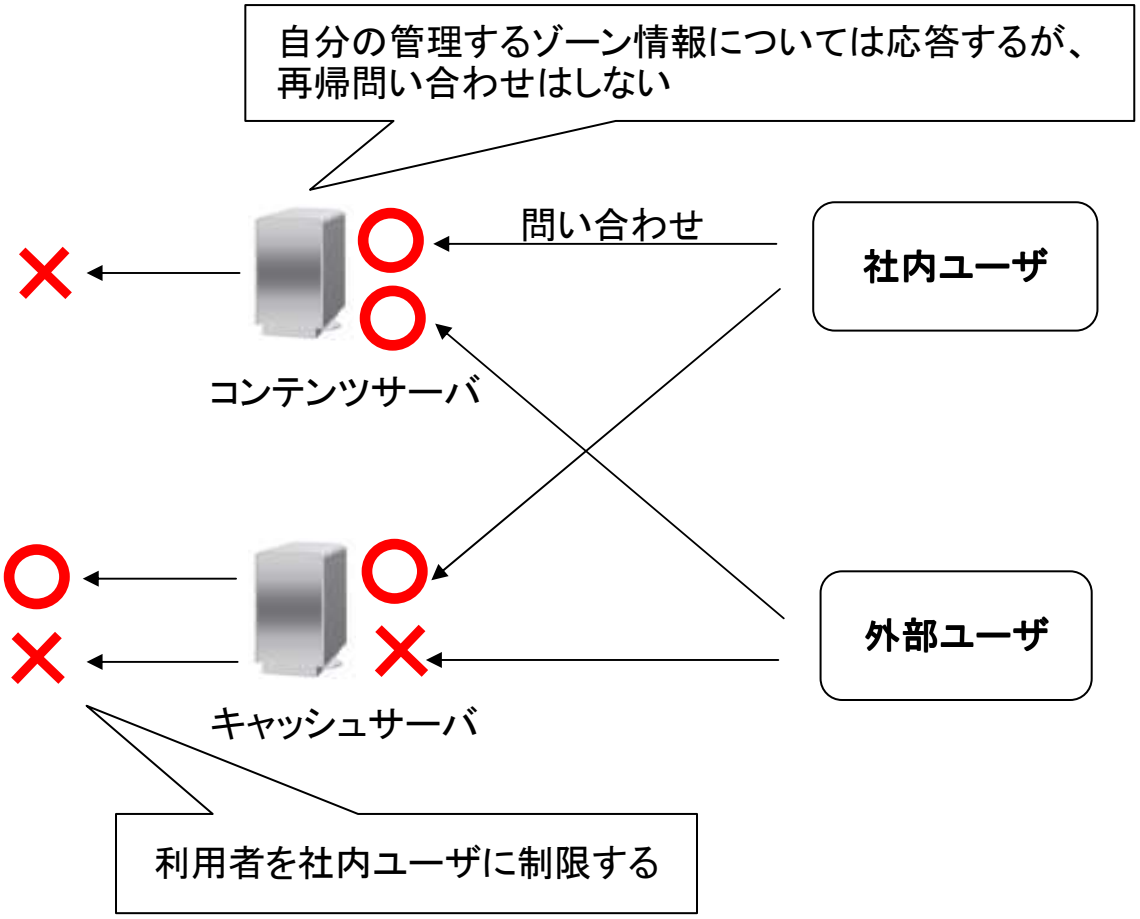
コンテンツサーバのnamed.conf

```
options {
#  再帰問い合わせを禁止
        recursion no;
 };
```

キャッシュサーバのnamed.conf

```
options {
#  再帰問い合わせの制限
      allow-recursion {
              IPアドレス/Prefix長;
              ACL名;
      };
};
```

※ 先頭の#は、コメント行を示す



日本語ドメイン名

日本語ドメイン名とは、その名の通りドメインに日本語を使用した国際ドメイン名のこと。
クライアント側は、日本語ドメイン対応ブラウザが必要。
DNSサーバ側では、日本語ドメインをPunycode表記に変換した文字列でドメイン設定をする。

クライアント側ブラウザ

③ 問い合わせ
www.xn--88j1au9swd7cpd0ttg6748cehdm06m.jp ？

④ 回答
192.168.24.54

DNSサーバ側

named.confファイル内

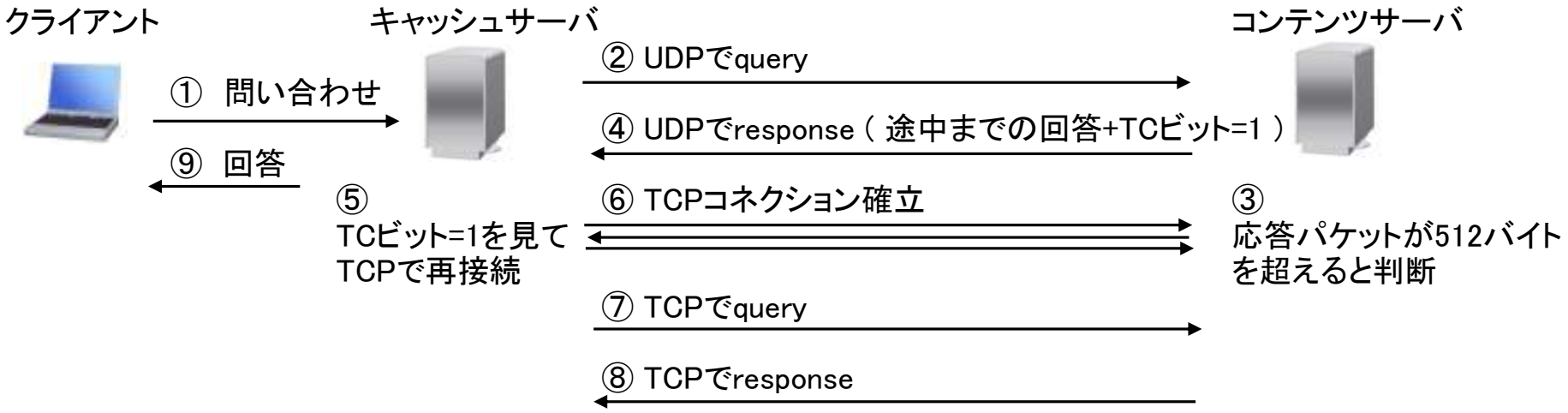
```
zone "xn--88j1au9swd7cpd0ttg6748cehdm06m.jp" IN {
        type master;
        file "zonefile/xn--88j1au9swd7cpd0ttg6748cehdm06m.jp.zone";
}
```

xn--88j1au9swd7cpd0ttg6748cehdm06m.jp.zoneファイル内

```
www       A   192.168.24.54
```

TCPフォールバック

DNSにおけるTCPフォールバックとは、パケットのサイズ（IPとUDPのヘッダを除く）が512バイトを超える場合に、TCPに切り替えて再度問い合わせる仕組みのこと。
通常、DNSへの問い合わせ応答にはUDPを使うが、パケットサイズはデフォルトで512バイト以下と決まっている。
TCPに切り替えるキックは、問い合わせ先DNSサーバが応答パケットにTCビット=1をセットすることから始まる。
最初からTCPで通信をすることはできない。



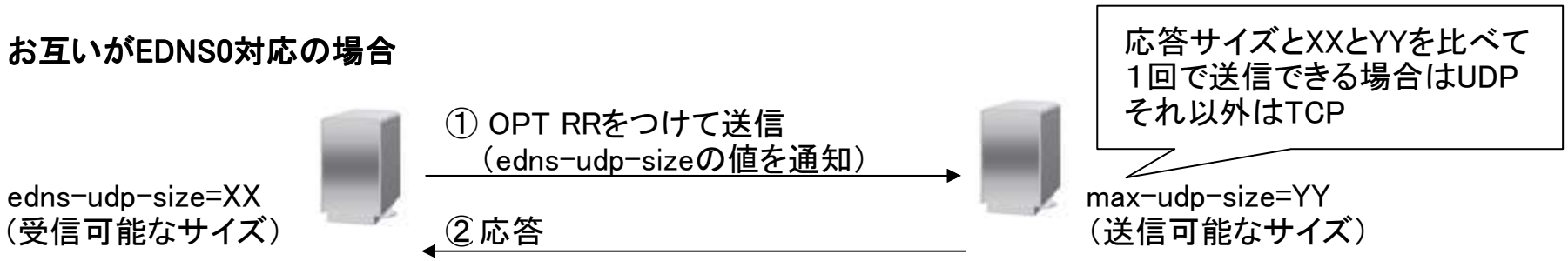
digコマンドでみるTCPフォールバック

$/usr/local/bin/dig @192.36.144.107 se. any ;; Truncated, retrying in TCP mode. （省略） ;; Query time: 306 msec ;; SERVER: 192.36.144.107#53(192.36.144.107) ;; WHEN: Tue Feb 23 14:32:02 2010 ;; MSG SIZE rcvd: 2688	TC=1を受け取ったので、TCPで再試行 受信パケットのメッセージサイズは、2688バイト

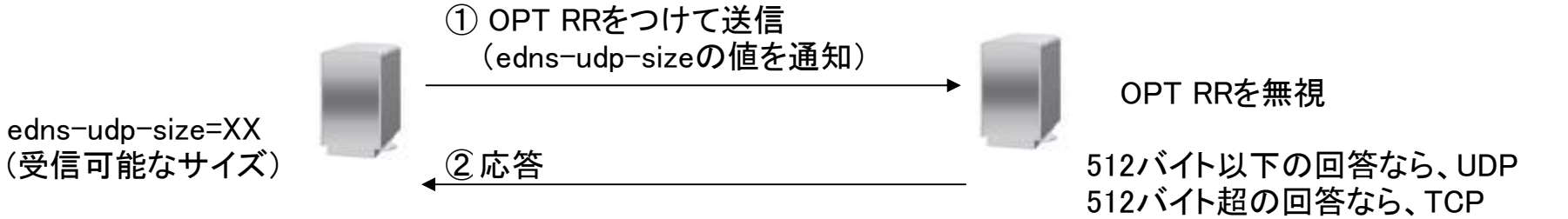
EDNS0

DNSの規格に対する拡張バージョン0のこと。
本来、DNSパケットサイズが512バイトを超える場合はTCPフォールバックが発生し、TCP通信に切り替わるが、
512バイト超の大きなDNSデータでもUDPで取り扱えるようにするための仕組み。
ただし、お互いがEDNS0に対応していないと利用できず、その場合は通常のUDP ⇒ TCPフォールバックとなる。
BIND9以降であれば、デフォルトで有効となっている。

お互いがEDNS0対応の場合



EDNSOでない場合



※ EDN0を実装していないサーバは、OPT擬似レコードを無視し、EDNS0を実装しているがサポートしていない
サーバは、エラーを返すようになっている
※ max-udp-sizeを超える場合はTCPフォールバックが発生する。

EDNS0の設定

BIND9以降ではデフォルトで有効であるため、特に設定をしていなくても利用できる。

```
options {
        edns-udp-size 数値;
        max-udp-size 数値;
};
```

edns-udp-size 数値;
EDNS0のUDPサイズ(バイト)を通知するサイズを指定する
有効な数値は、512バイトから4096バイト
デフォルトの数値は、4096バイト

```
server IPアドレス/プリフィックス {
        edns yes | no ;
        edns-udp-sise 数値;
        max-udp-size 数値;
};
```

max-udp-size 数値;
回答時に送信するEDNS0のUDPメッセージの最大値を指定する
有効な数値は、512バイトから4096バイト
デフォルトの数値は、4096バイト

serverステートメント
特定のDNSサーバを対象とする場合に作成する

edns yes | no ;
指定のリモートサーバとの通信時にEDNSを使用するかどうかを指定する
デフォルトはyes

digコマンドでみるEDNS0

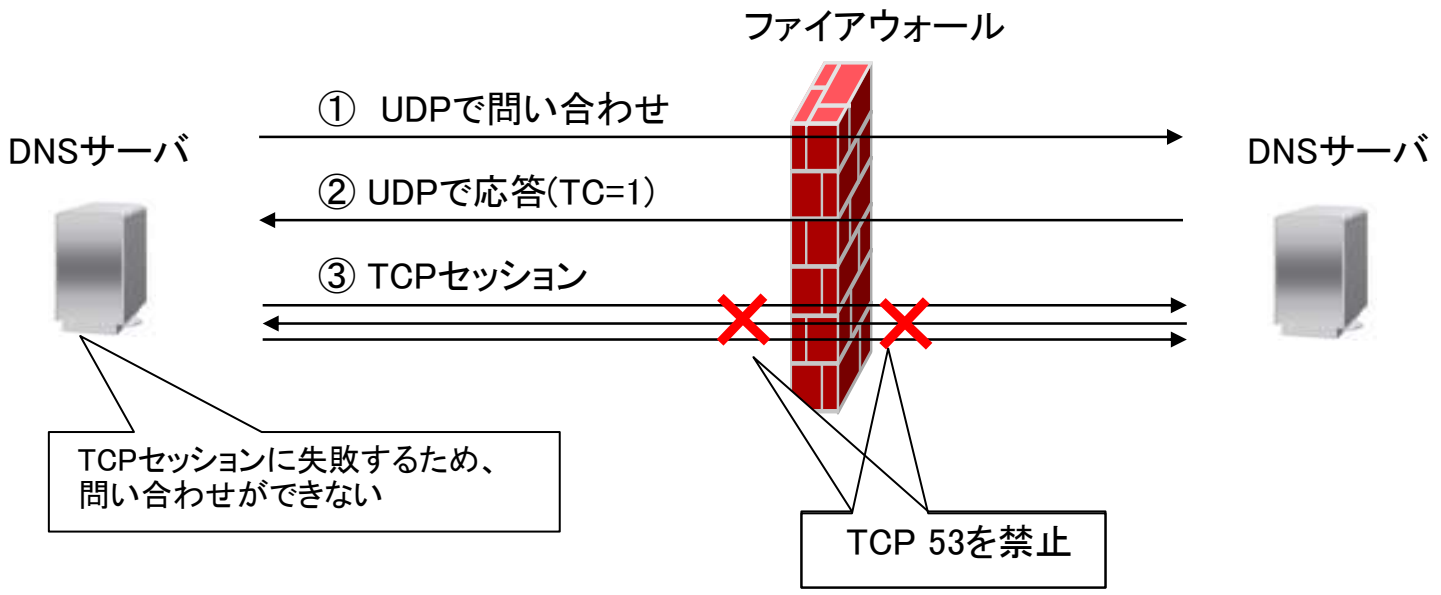
$/usr/local/bin/dig @192.36.144.107 se. any +bufsize=3000	+bufsize=XXで受信可能なサイズを通知
	TCPではないので
(省略)	「;; Truncated, retrying in TCP mode.」
;; Query time: 306 msec	が表示されていない
;; SERVER: 192.36.144.107#53(192.36.144.107)	
;; WHEN: Tue Feb 23 14:32:02 2010	
;; MSG SIZE rcvd: 2699	受信パケットのメッセージサイズは、2699バイト

DNSパケットサイズと問題点

DNSのやり取りでは、通常UDPを利用する。
パケットサイズは512バイト以下とし、それを超える場合はTCPに切り替える仕組みとなっている。
ただし、EDNS0に対応していれば、パケットサイズが512バイトを超えてもUDPで扱える。

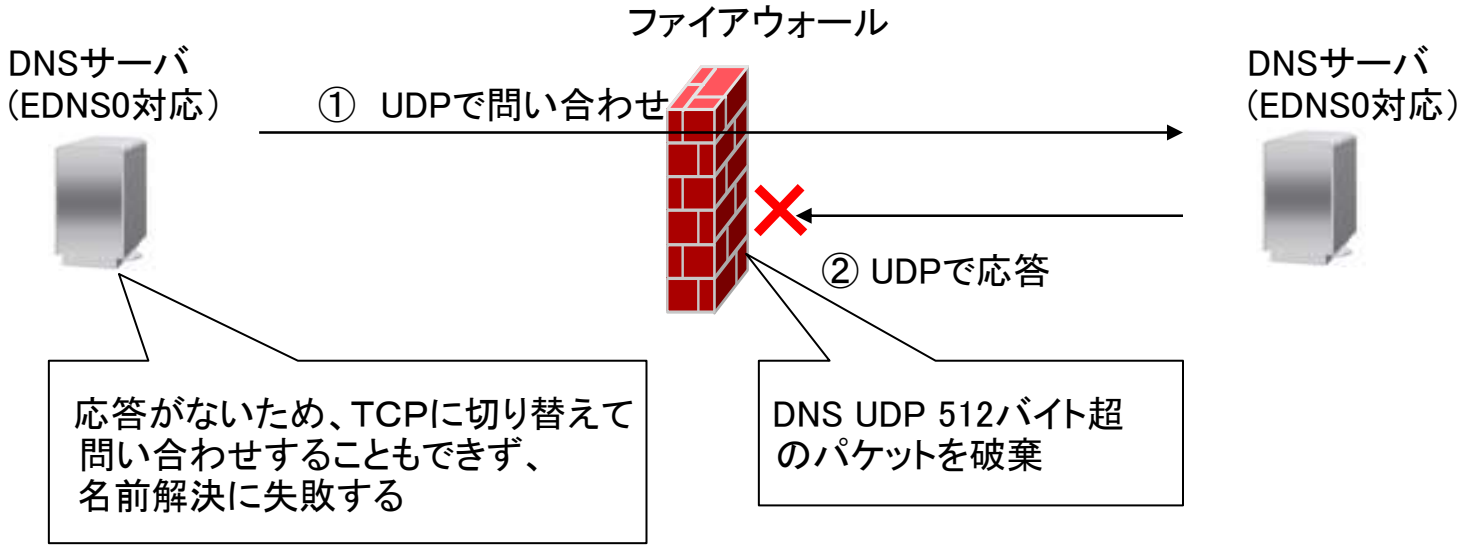
問題点 1
TCP 53番ポートを閉鎖しているFW等がある場合、TCPフォールバックに切り替わった後にTCPセッションがはれないために名前解決に失敗する。

解決方法
双方向で、送信元ポート番号がTCP 53番のポートを開放する。



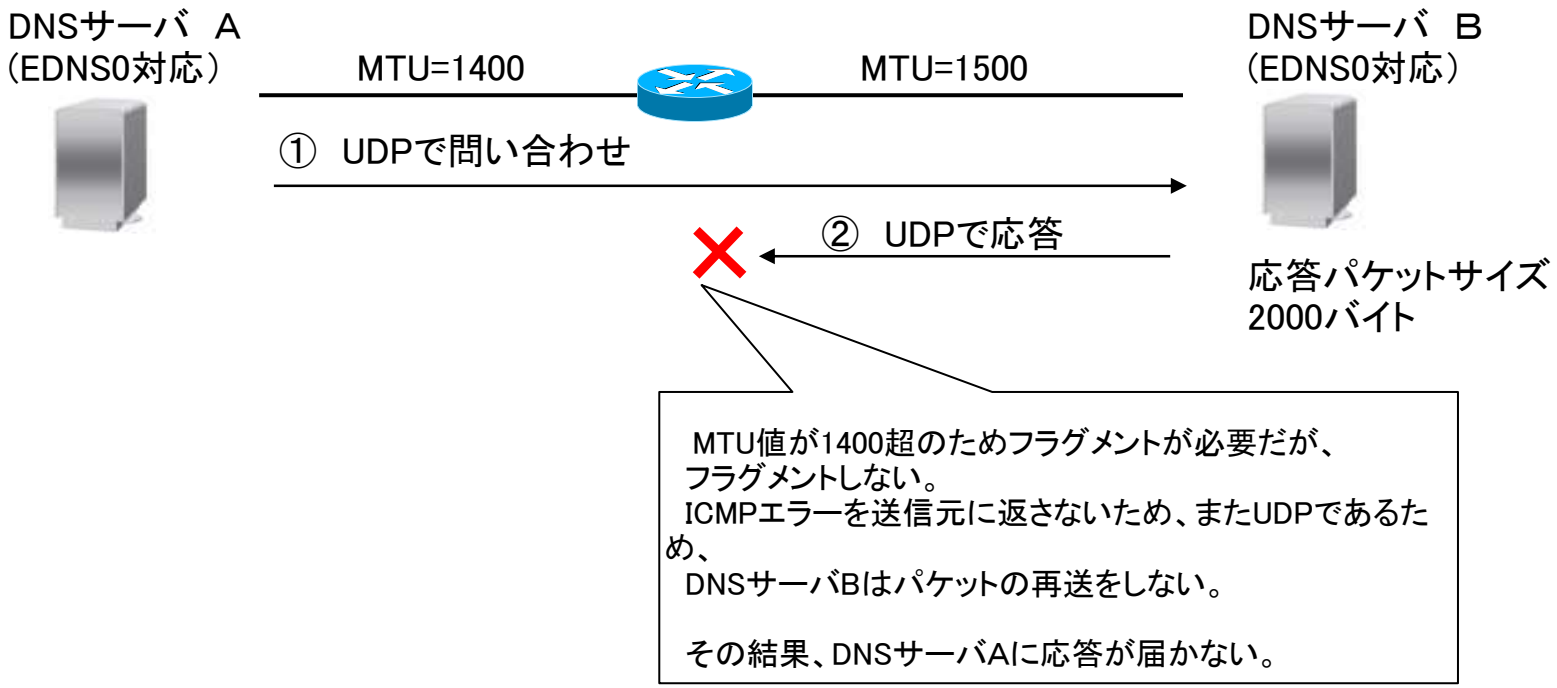
問題点2
お互いがEDNS0対応している場合512バイトを超えるUDPパケットを扱うことができるが、古い装置やOSでは、512バイトを超えるUDPパケットを破棄するものがある。

解決方法
EDNS0に対応したバージョンに上げる。 または、装置の取替え。



問題点3
途中にICMPエラーを返さなかったりフラグメントしない装置がある場合、MTU問題により名前解決ができない。

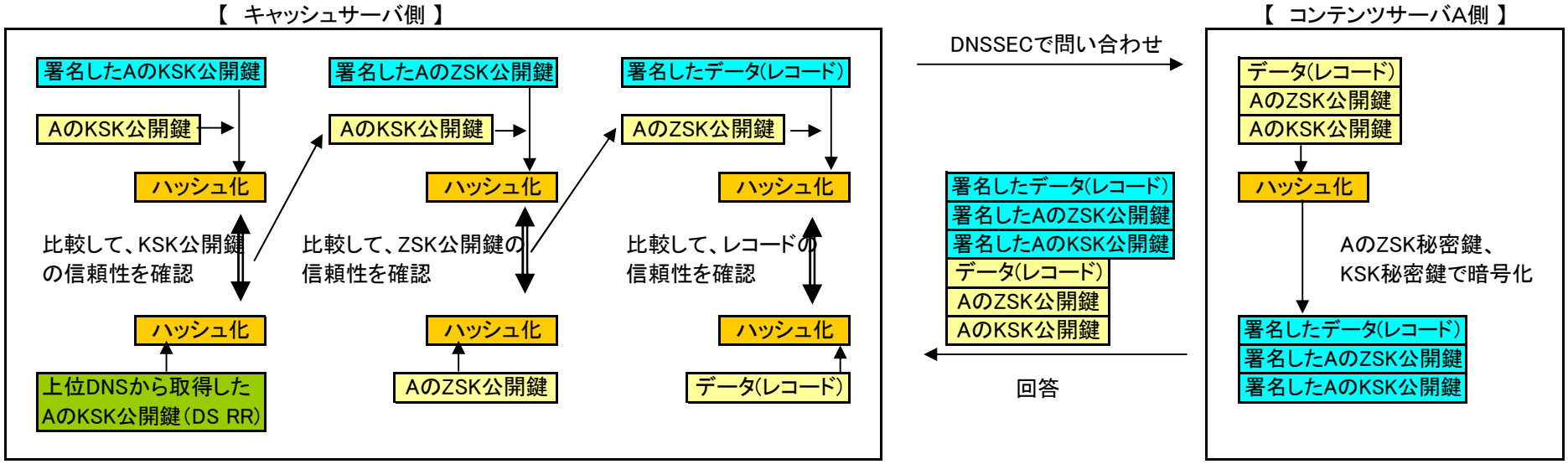
解決方法
IMCPエラーを返すようにする。またはフラグメントを許可する。



DNSSEC

DNSのリソースレコードに電子署名を付加する事により、DNSの応答パケットの完全性を証明するための仕組み。
署名が正しいことを証明するために、ゾーンを署名するZSK公開鍵と、そのZSK公開鍵と自身を証明するKSK公開鍵の2つの鍵が使用される。

DNSSECの仕組み



信頼の連鎖

上図では、AのKSK公開鍵を証明する方法がない。
そこで、AのKSK公開鍵を上位のDNSサーバに登録・署名してもらうことにより、そのKSK公開鍵が正しいことを証明し、さらに上位のDNSサーバは、自身のKSK公開鍵を上位のDNSサーバに登録・証明してもらう。
最終的にはルートDNSサーバにたどり着く。
ルートDNSサーバのKSK公開鍵だけは、あらかじめキャッシュサーバに登録しておく。

DNSSECの設定(キャッシュサーバの場合)

キャッシュサーバで必要な設定は2つ。

① DNSSECを有効にする

```
# vi named.conf
options {
dnssec-enable   yes | no ;        # DNSSECを有効にするかどうか　BIND9.5以降は、デフォルトでyes
dnssec-validation yes | no ;      # DNSSECによる検証を行うかどうか BIND9.5以降は、デフォルトでyes
};
```

例） # vi named.conf
options {
dnssec-enable   yes ;
dnssec-validation yes ;
};

② トラストアンカーを設定する

DNSSECを構成する最上位のゾーンのKSK公開鍵を登録する。
すでにルートDNSがDNSSECに対応しているので、今回はルートDNSのKSK公開鍵を登録する。
named.confの設定では、managed-keysステートメントまたはtrusted-keysステートメントを使用する。

トラストアンカー(KSK公開鍵)を取得する

```
# dig +noall +answer DNSKEY . > 出力ファイル        # ルートDNSのDNSKEYレコードをファイルに保存
```

例） # dig +noall +answer DNSKEY . > root-dns.key　　# ファイルの中にZSK公開鍵とKSK公開鍵がある
「DNSKEY 256 3 8」がある行は、ZSK公開鍵を示す
「DNSKEY 257 3 8」がある行は、KSK公開鍵を示す

トラストアンカーを設定する

```
managed-keys {                                  # 登録したトラストアンカーのKSK公開鍵で署名された
"信頼するゾーン" initial-key 257 3 8               新しいトラストアンカーであればそれを信用する。
"KSK公開鍵";                                      BIND9.7以上で対応
};

trusted-keys {                                  # managed-keyとほぼ同じだが、KSK公開鍵の変更
"信頼するゾーン" 257 3 8 "KSK公開鍵";               があれば手動で変更する必要がある。
};                                              BIND9.6まではこちらしか使えない。
```

```
例# vi named.conf
      managed-keys {                                    # ルートDNSゾーン( . )に対してトラストアンカーを設定
            "." initial-key 257 3 8 "AwEAAagAIKlVZrpC6I
      (省略)
      };
```

named.conf 設定例

```
options {
       dnssec-enable yes;
       dnssec-validation yes;
};

trusted-keys{
      "." 257 3 8 "AwEAAagAIKlVZrpC6I
(省略)
mqrAmRLKBP1dfwhYB4N7knNnulq QxA+Uk1ihz0=";
};
```

DNSSECの設定(コンテンツサーバの場合)

コンテンツサーバの場合は、キャッシュサーバで行った2つの設定以外に、管理するゾーンの署名が必要。

① ZSK鍵とKSK鍵の作成

```
# dnssec-keygen -a 鍵方式 -b ビット数 -r 鍵作成に利用するファイル -n ZONE ゾーン名
                                                    # ZSK鍵(公開鍵と秘密鍵)の作成
# dnssec-keygen -a 鍵方式 -b ビット数 -r 鍵作成に利用するファイル -f KSK -n ZONE ゾーン名
                                                    # KSK鍵(公開鍵と秘密鍵)の作成
```

例） # dnssec-keygen -a RSASHA1 -b 1024 -r /dev/urandom -n ZONE example.jp
dnssec-keygen -a RSASHA1 -b 1024 -r /dev/urandom -f KSK -n ZONE example.jp

（生成されるファイル）

```
Kexample.jp.+005+{識別番号}.key      # ZSK公開鍵
Kexample.jp.+005+{識別番号}.private  # ZSK秘密鍵
Kexample.jp.+005+{識別番号}.key      # KSK公開鍵
Kexample.jp.+005+{識別番号}.private  # KSK秘密鍵
```

※ 公開鍵と秘密鍵の識別番号は同じになる
ZSK鍵かKSK鍵かはそれぞれの公開鍵の中身を見ないと分からない
ZSK鍵は「～DNSKEY 256～」、KSK鍵は「～DNSKEY 257～」となる

② 作成したZSK公開鍵とKSK公開鍵をゾーンファイル内に追記する
追加した後、ゾーンファイルのシリアル番号を変更する

```
# cat ①で作成されたZSK公開鍵ファイル >> ゾーンファイル
# cat ①で作成されたKSK公開鍵ファイル  >> ゾーンファイル
```

例） # cat Kexample.jp.+005+31596.key >> example.jp.zone
cat Kexample.jp.+005+34663.key >> example.jp.zone

③ 作成したZSK秘密鍵を使用してゾーンファイルに署名を行う
同時にDSSETのファイル（DSレコードが記述されたファイル）も作成される

```
# dnssec-signzone -o ゾーン名 ゾーンファイル 出力ゾーンファイル
```

例） # dnssec-signzone -o example.jp example.jp.zone example.jp.zone.signed

（生成されるファイル）　example.jp.zone.signed　# 署名されたゾーンファイル
dsset-example.jp.　# DSSETファイル

④ ③で作成した署名済みゾーンファイルをnamed.confに登録する

```
# vi named.conf
zone ゾーン名 {
        type master;
//       file "署名前のゾーンファイル";                    # 署名前のゾーンファイルは削除するか
        file "③で作成したゾーンファイル";                   コメントアウトする
};
```

例） # vi named.conf
```
zone example.jp {
        type master;
//       file "zonefile/example.jp.zone";
        file "zonefile/example.jp.zone.signed";
};
```

⑤ ③で作成したDSSETの内容(DSレコード)を上位のDNSサーバに登録してもらう

例） # more dsset-example.jp.
example.jp. IN DS 34663 5 1 213B(省略)2E013F919
example.jp. IN DS 34663 5 2 D82A(省略) 5B50DC27

named.conf 設定例

```
options {
        dnssec-enable yes;
        dnssec-validation yes;
};

trusted-keys {
        "." 257 3 8 "AwEAAagAIKlVZrpC6I
(省略)
mqrAmRLKBP1dfwhYB4N7knNnulq QxA+Uk1ihz0=";
};

zone example.jp {
        type master;
//       file "zonefile/example.jp.zone";
        file "zonefile/example.jp.zone.signed";
};
```

DNSSECの動作確認

digコマンドを使って、DNSSECによる問い合わせの確認をすることができる。

例） orgドメインのSOAレコードについて、DNSSECを使って問い合わせた結果

# dig @localhost org. soa +dnssec	**+dnssecオプションをつける** → EDNS0も自動的に有効
（省略） ;; flags: qr rd ra **ad**; QUERY: 1, ANSWER: 2, AUTHORITY: 7, ADDITIONAL: 1	**DNSSECの問い合わせができている場合はadビットが立つ**
;; OPT PSEUDOSECTION: ; EDNS: version: 0, flags: do; udp: 4096 ;; QUESTION SECTION: ;org. IN SOA	EDNS0はデフォルトで edns-udp-sise=4096で問い合わせる
;; ANSWER SECTION: org. 451 IN SOA a0.org.afilias-nst.info. noc.afilias-nst.info. （省略） org. 451 IN RRSIG SOA 7 1 900 （省略） 20100903051622 37812 org. （省略） 9sqxzy2h1uW206l/3seemCplRbR0jPM86QzsMNLmt4ZTrRvycChXXYNa UFc=	SOAレコードに対するRRSIGが付与
;; AUTHORITY SECTION: org. 451 IN NS d0.org.afilias-nst.org. （省略） org. 451 IN NS a2.org.afilias-nst.info. org. 451 IN RRSIG NS 7 1 86400 （省略）20100901144517 37812 org （省略） /GHkkjtLZrDZEMN SgXgMTVi0AYwYMq3785j/91glI2pxAyBXR5J37VP0WSWOkdsDCq5KJFZ KB0=	NSレコードに対するRRSIGレコードが付与

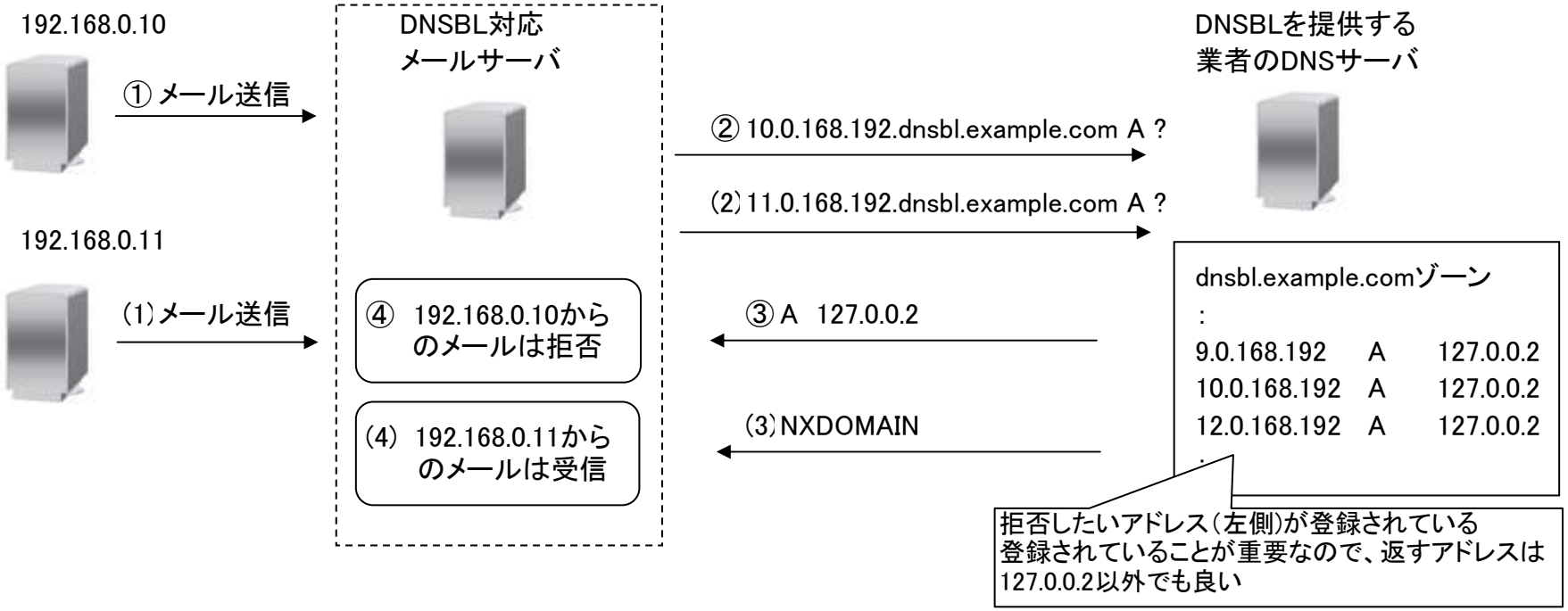
DNSBL(DNS Black List)

DNSBLとは

スパムメールの送信や無制限に中継を行っているホストなど、拒否したいIPアドレスを集めたリスト。
DNSの仕組みを利用して情報を提供している。
メールサーバや掲示板、ブログサイトなどでDNSBLが利用されている。

DNSBLは、そうした仕組みを利用したデータベースの総称である。
また、DNSBLの正式名称は、DNS Black Listというわけではない。
ドメインを対象にしたRHSBや、2ちゃんねるが採用しているプロキシを対象にしたBBQなどもある。

DNSBLの基本的な仕組み



sendmailでの設定例

自分が使用したいDNSBLのデータベースをFEATUREの部分に追加する。

① sendmail.mcに追加

```
# vi /etc/mail/sendmail.mc
FEATURE(dnsbl,`all.rbl.jp')dnl
```

② sendmail.mcからsendmail.cfを作成

```
# m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

③ sendmail.cfの再読み込み

```
# service sendmail reload
```

DNSBLの問題点

以下の理由で、最近ではDNSBLの利用に否定的な立場の人が多い。

- 動的IPの場合、他のユーザがスパムを発信したためにアドレス全体がDNSBLに登録されたり、登録された動的IPをたまたま割り当てられた無関係なユーザが迷惑を被ることがある。
- DNSBLの管理がきちんとされていないデータベースがある。
- DNSに無駄な負荷がかかっている。

DNSBLに登録されているか確認する方法

DNSBLを管理しているサイトや、それらをまとめたサイトで確認ができる。(コマンドでも可能)

例） 主要なブラックリスト一覧が記載されたHP　http://www.dnsbl.info/dnsbl-list.php
いろいろと問題の多いSORBSのHP　http://www.au.sorbs.net/
2chが運営するBBQ　http://bbq.uso800.net/