DNSサーバーの安全な設定

民田雅人
minmin@jprs.co.jp
株式会社日本レジストリサービス
DNS DAY – Internet Week 2003

サーバーで安全な設定とは

- 正しい情報を正しく提供する
  - 不確かな情報を提供したりしない
    (安全というより正しい設定)
- サービス経由で侵入されない
  - 万が一侵入されても被害を最小限にする

DNSの復習

- DNS(Domain Name System)は、
  サーバーとクライアントから成り立つ
- ネームサーバー
  - 専用のサービスプログラム
    - named(BIND), tinydns(djbdns), MicrosoftDNS(Windows), etc…
- リゾルバ
  - ライブラリ
  - サービスプログラム

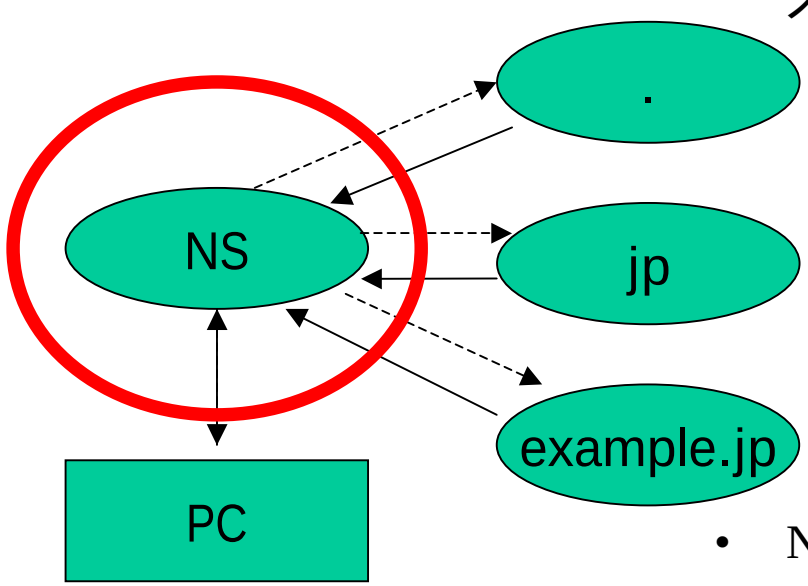
2種類のネームサーバー(1)

- クライアントの問い合わせ(再帰的問い合わせ)に答える
  - www.example.jpのIPアドレスを検索する要求
  - IPアドレスが10.20.30.40のホスト名を検索する要求
- 回答を持っていない場合、DNSの検索を行う
  - 結果をキャッシュして同一の問い合わせに備える
    →トラフィックと負荷の削減
- 「キャッシュサーバー」と呼ぶ
  - 「フルリゾルバ」とも呼ぶ

キャッシュサーバー

- PCなどのクライアントに設定する
- マニュアルで設定する
  - /etc/resolv.confでnameserver行に設定
  - ダイアルアップのネームサーバーの設定
- 自動で設定する
  - DHCPサーバーでクライアントに配布
  - PPPでクライアントに配布

www.example.jpの IPアドレスの検索



- PCからNSへwww.example.jpのIPアドレスの問い合わせ
  - NSが“.”(ルート)のnsにIPアドレスを問い合わせ
  - “.”のnsがjpのnsを返答
  - NSがjpのnsへIPアドレスを問い合わせ
  - jpのnsがexample.jpのnsを返答
  - NSがexample.jpのnsへIPアドレスを問い合わせ
  - example.jpのnsがwww.example.jpのIPアドレスを返答
- NSがPCへIPアドレスを返答

ネームサーバー(nsと略す)

問い合わせ

返答

2種類のネームサーバー(2)

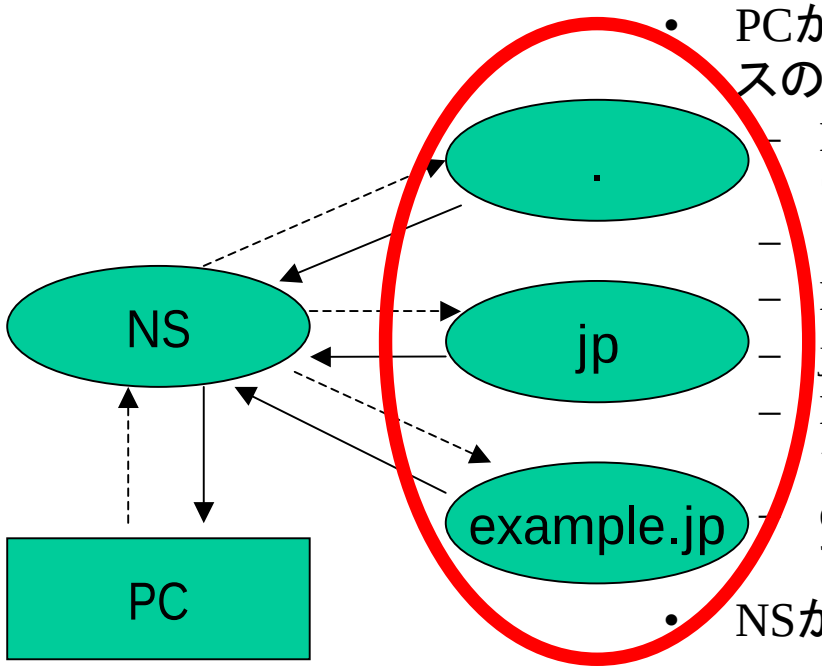
- 管理しているドメインについての
  問い合わせ(非再帰的問い合わせ)に答える
  - www.example.jpのIPアドレスは?
  - 10.20.30.40のホスト名は?
  - 主にキャッシュサーバーからの問い合わせ
- 問い合わせが管理外のドメインの場合、
  回答しない
  - エラーを返す or 何も返さない
- 「コンテンツサーバー」と呼ぶ

コンテンツサーバー

- 上位ドメインに登録し管理を委任するネームーサーバー

```
% dig @a.dns.jp example.jp ns
;; ANSWER SECTION:
example.jp.  1D IN NS  ns0.example.jp.
example.jp.  1D IN NS  ns1.example.jp.
;; ADDITIONAL SECTION:
ns0.example.jp. 1D IN A xx.xxx.xxx.xx
ns1.example.jp. 1D IN A yy.yyy.yyy.yy
```

www.example.jpの IPアドレスの検索



- PCからNSへwww.example.jpのIPアドレスの問い合わせ
  - NSが“.”(ルート)のnsにIPアドレスを問い合わせ
  - “.”のnsがjpのnsを返答
  - NSがjpのnsへIPアドレスを問い合わせ
  - jpのnsがexample.jpのnsを返答
  - NSがexample.jpのnsへIPアドレスを問い合わせ
  - example.jpのnsがwww.example.jpのIPアドレスを返答
- NSがPCへIPアドレスを返答

ネームサーバー(nsと略す)

問い合わせ

返答

コンテンツサーバーなのに

- ある組織のネームーサーバーである ns.example.jpへ、その組織と関係ない www.example.comのIPアドレスを問い合せると 回答がある
  - dig @ns.example.jp www.example.com a
- コンテンツ・キャッシュサーバーを兼用し、 適切なアクセス制限ができていない

第三者による キャッシュサーバーの不正利用

- 不正にドメインの検索を大量に行える
  - 負荷の増大
  - キャッシュメモリの増大
    - BIND9ならキャッシュメモリに制限をかけられるのでまだまし
  - プログラムの穴を突く可能性もありうる
    - キャッシュ汚染の可能性もある
- いずれもDoS攻撃につながる
  - コンテンツサーバーに影響があると致命的になる
- 普通に使われて通常のドメインを検索するなら おそらくほとんど問題は発生しない

アクセス制限を考慮したBINDの設定例

- 再帰的検索は管理対象ネットワークのみに制限
- 管理するゾーンへの問い合わせは何処からでも

```
options {
    ...
    recursion yes ;
    fetch-glue no ;
    allow-query {
        localhost ;
        10.0.0.0/8 ;
    } ;
    ...
};
```

```
zone "." {
    type hint;
    file "named.root" ;
};
zone "0.0.127.IN-ADDR.ARPA" {
    type master ;
    file "localhost.rev" ;
};
zone "example.jp" {
    type master ;
    file "example.jp.zone" ;
    allow-query { any; };
};
```

自組織のネットワークが 10.0.0.0/8 の例

コンテンツサーバーと キャッシュサーバーは別に運用する

- キャッシュ汚染からコンテンツサーバーを守る
  - よりセキュアな設定に
  - 一方のトラブルで他方が巻き込まれるのを防ぐ
- BINDはコンテンツ・キャッシュの 明示的な区別が無い
  - namedで両方を兼用
  - WindowsのDNSサーバーもBINDと同様
  - BIND, WindowsDNSは設定で分離可能
- djbdnsではもともと別のプログラムとして実装
  - tinydns(コンテンツ)とdnscache(キャッシュ)で兼用不可

BINDでのコンテンツサーバー

- named.confには自組織関連のゾーンを記述
- recursion no;
- fetch-glue no;
  - BIND9では常にno
- hint情報不要(zone “.”)
- セカンダリの場合、zoneの記述部分で、マスターから転送するように設定する

```
options {
        ...
        recursion  no;
        fetch-glue no;
        ...
} ;
zone "example.jp" {
        type master ;
        file "example.jp.zone" ;
} ;
```

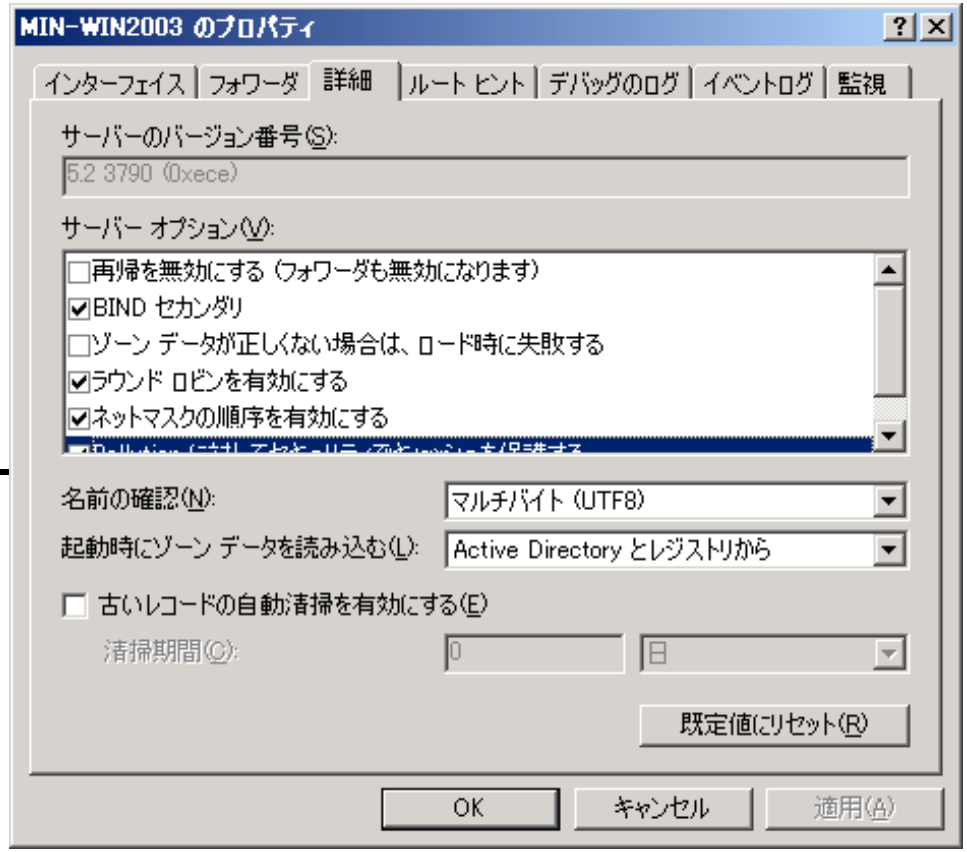
BINDでのキャッシュサーバー

- ゾーンとしては左の2つのみ
  - 自組織関連ゾーンのセカンダリーを設定してもよい
- recursion yes;
- hint情報が必要
- allow-queryでアクセス制限して第三者に不正に利用させない
- 127.0.0.1 (localhost)の情報も加える。
  - ::1もお忘れなく。
- 1台のサーバーでもBINDを複数起動することは可能
  - 付録参照

```
options {
        ...
        recursion yes;
        fetch-glue no;
        allow-query {
           10.0.0.0/8 ;
        };
        ...
};
zone "." {
        type hint;
        file "named.root";
};
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};
```

WindowsのDNSサービスの場合

- DNSのプロパティの「詳細」タブ
  - 「再帰を無効にする」
    - チェックするとコンテンツ専用サーバーになる
- DNSサーバーの設定だけではアクセスコントロールはできない
  - ルータによるフィルターまたはWindows用BINDに変更



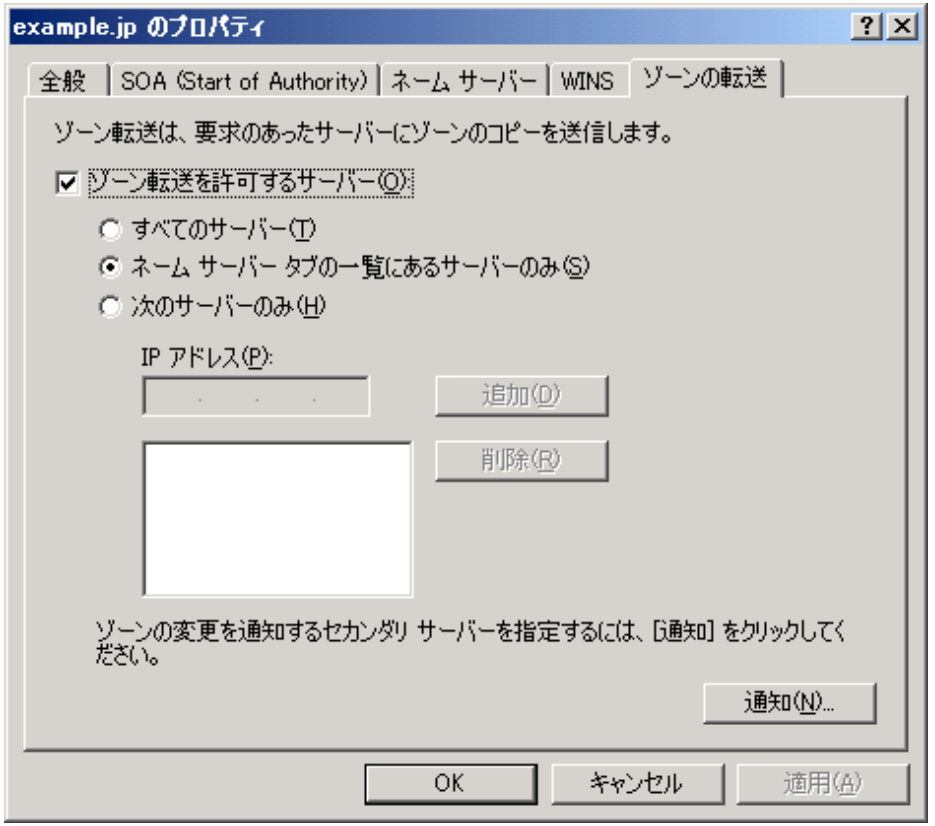
ゾーン転送可能なホストを制限する

- セカンダリー以外には転送できないように
  - ゾーン転送はDNS的に重い処理なので
    サービス不能攻撃の原因になりかねない
- BINDの場合、optionsやzoneにallow-transferで
  セカンダリーサーバーのIPアドレスを記述。

```
zone “example.jp” {
    ...
    allow-transfer { x.x.x.x ; y.y.y.y ; };
    ...
};
```

WindowsのDNSサービスでのゾーン転送制限

- 該当ゾーンのプロパティで設定する
  - デフォルトでは、NSレコードに指定したホストのみ転送を許可する



BINDのバージョン

- 2003年11月28日現在のBIND
  - BIND 9 系　　Version 9.2.3
    2003年10月23日リリース
  - BIND 8.系　　Version 8.4.3
    2003年11月26日リリース
- 過去のバージョンでは侵入される危険がある
  - いうまでもないことですが…

万が一named経由で侵入されたときへの備え

- rootとは別のnamed専用ユーザーを用意しその権限で稼動するようにする
  - named –u <user> …
    - /var/run/named.pidなどのオーナーに注意
- namedをchroot環境で稼動させて、アクセスできるファイルを制限する
  - named –t <chrootディレクトリ> …
  - BIND9での設定例

    http://www.unixwiz.net/techtips/bind9-chroot.html
  - djbdnsはchroot環境下で動作する

セカンダリーネームサーバー

- 用意するなら違うネットワークに配置する
  - 負荷分散目的なら同一ネットワークもありうる
- セカンダリーの運用状況は本当に大丈夫か?
  - 第三者(接続先プロバイダ等)に任せるなら十分信頼できるところへ
  - プライマリがセキュアでも、セカンダリが…
- セカンダリサーバーの情報が正常かどうかを、定期的に確認する
  - ある日気づくと…ということの無いように

ルータでのacl や IDS

- ネームサーバーへのacl
  - 設定するのはかまわないけど…
  - 動作が妨げられない程度に
  - DNSは条件によってはTCPも利用する…
- IDS
  - 正常なパケットを侵入と検出したりしない
    - 誤検出によって、しなくてもいい問い合わせ
  - 生半可な設定は世間へ迷惑
  - 設定した本人も余計なコストがる

DNSパケットの横取り対策

- ネームサーバーと同一LANセグメントでパケットを覗けば、横取りはたやすい。
  - ネットワーク的にも近いため、
    正しい回答より先に嘘を返せる確立が高まる
- 「スイッチングハブにすればパケットは覗けない」と安心するのは大きな誤り。
  - ARP PoisoningとかARP Spoofingと呼ばれる手法
- Googleで検索
  - ARP Poisoning　　7,610件
  - ARP Spoofing　　57,800件

ARP Poisoning (1/2)

- スイッチングハブにhostA(キャッシュサーバー)、hostB(管理の甘いサーバ)、gw(ルータ)が接続

	IPアドレス	MACアドレス
gw	10.10.10.1	0:1:1:1:1:1
hostA	10.10.10.2	0:2:2:2:2:2
hostB	10.10.10.3	0:3:3:3:3:3

- アタッカーはhostBに侵入し楽々root権限を入手
- hostBから偽のARP応答を送る

```
hostA へ 10.10.10.1 のMACaddrは 0:3:3:3:3:3
gw   へ 10.10.10.2 のMACaddrは 0:3:3:3:3:3
```

- hostAとgwのARPテーブルが書き換わる
  - すべての通信はhostBを経由するようになる

ARP Poisoning (2/2)

- hostBでは入ってくるパケットを覗き、そのまま本来のIPアドレスへ転送する
  - Layer2的にはhostB宛なので、ネットワークインターフェースをプロミスキャスにする必要も無い
- ARP Poisoningされても通常の通信は問題なく行えるため気づきにくい
  - OSによってはARPテーブルが変化するとsyslogに残る

```
arp: 10.10.10.1 moved from 00:01:01:01:01:01 to 00:03:03:03:03:03 on em0
arp: 10.10.10.1 moved from 00:03:03:03:03:03 to 00:01:01:01:01:01 on em0
arp: 10.10.10.1 moved from 00:01:01:01:01:01 to 00:03:03:03:03:03 on em0
arp: 10.10.10.1 moved from 00:03:03:03:03:03 to 00:01:01:01:01:01 on em0
```

ARP Poisoning対策

- 同一セグメントに繋がっているホストを
すべて正しく管理
  - セキュリティホールを残さないこと
  - パスワード管理も正しく行う
- ARPテーブルをスタティックに登録する
  - 手間はかかるが、管理したマシンしか接続できなくなるため、セキュリティ的には強固になる。

まとめ

- 古き良き時代は過去の話
  - メールサーバーのオープンリレーと同様…
- 十分なアクセス制限
- 十分なセキュリティ対策
- 今一度、自分の管理してるネームサーバーとファイアウォール周りの点検をしてみましょう

付録

1台でキャッシュサーバーとコンテンツサーバーを運用(1/3)

- namedプロセスを2つ起動する
  - 但しBIND9はv6 を有効にすると1プロセスのみ listen-on-v6 {any;}; のみ機能 → 将来修正される(?)
- コンテンツサーバー用/etc/named.conf

```
options {
    ...
    recursion  no;
    fetch-glue no;
    listen-on { 10.10.10.1 ; } ;
    ...
} ;
```

- listen-onでサーバーのIPアドレスのみ
- /etc/resolv.confでは“nameserver 127.0.0.1”

1台でキャッシュサーバーとコンテンツサーバーを運用(2/3)

- キャッシュサーバー用/etc/cache.conf
  - named –c /etc/cache.conf で起動

```
options {
    ...
    pid-file     "/var/run/cache-named.pid" ;
    listen-on { 127.0.0.1 ; } ;
    ...
};
controls {
    unix "/var/run/cache-ndc" perm 0600 owner 0 group 0;
} ;
```

- 127.0.0.1だけなのでアクセス制限は不要

1台でキャッシュサーバーとコンテンツサーバーを運用(3/3)

- dump-file, memstatistics-file, statistics-fileにも注意
  - 2つのnamedプロセスで上書きの可能性があるため一方を名前を変更する
  - 例　(BIND8の場合)

```
dump-file           "cache_dump.db" ;
memstatistics-file  "cache.memstats" ;
statistics-file     "cache.stats" ;
```

キャッシュサーバーに加えるべき逆引きゾーンの設定

- Private Address Space - RFC 1918
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- IPv4 Link-Local Address
  - Dynamic Configuration of IPv4 Link-Local Addresses
    - draft-ietf-zeroconf-ipv4-linklocal-07.txt
  - 169.254.0.0/16
- 特にISPのネームサーバー担当の方は是非!

キャッシュサーバーに加えるべき逆引きゾーンの設定例

- named.conf

```
zone "10.in-addr.arpa" {
   type master; file "dummy.zone"; };
zone "16.172.in-addr.arpa" {
   type master; file "dummy.zone"; };
.......
.......
zone "31.172.in-addr.arpa" {
   type master; file "dummy.zone"; };
zone "168.192.in-addr.arpa" {
   type master; file "dummy.zone"; };
zone "254.169.in-addr.arpa" {
   type master; file "dummy.zone"; };
```

- dummy.zone
  - SOAとNSを記述
  - 他は不要

```
$TTL 1D
@    IN SOA ns.example.jp.
                    root.example.jp. (
                1
                1H
                15M
                1W
                1D )
     IN NS    ns.example.jp.
```