DNS ブロッキングによる児童ポルノ対策ガイドライン

第２版

2012 年 11 月 2 日

安心ネットづくり促進協議会

調査研究委員会　児童ポルノ対策作業部会

ISP 技術者サブワーキンググループ

改訂履歴

版数	発行日	改訂履歴
第1版	2011年4月28日	初版発行
第2版	2012年10月●日	BIND Response Policy Zoneを使用したDNSブロッキングの記述を追加（ 6項 ） 商用DNS製品を使用したDNSブロッキングの概要説明を追加（ 9項 ）

1. 本ガイドラインの目的

2010 年 7 月の犯罪対策閣僚会議において策定された児童ポルノ排除総合対策において、「児童ポルノ掲載アドレスリストの迅速な作成・提供等実効性のあるブロッキングの自主的導入に向けた環境整備」「ISP による実効性のあるブロッキングの自主的導入の促進」が盛り込まれ、民間主導による検討が進められてきた。児童ポルノ掲載アドレスリスト（以下、アドレスリスト）の管理・作成団体については、民間により一般社団法人インターネットコンテンツセーフティ協会[1]がアドレスリスト管理団・作成団体として設立、選定され、2011 年 4 月 1 日より ISP 事業者や検索事業者等へのアドレスリストの提供が開始されることとなった。一方で、ISP においてもアドレスリスト提供に合わせてブロッキング実施に向けた検討を各社行ってきている。2010 年 6 月に ISP 技術者サブワーキンググループでとりまとめた報告書では、ブロッキングに関する ISP へのアンケート結果として、採用可能なブロッキング方式として回答のあった ISP の 4 割強が DNS を利用したブロッキング（以下、DNS ブロッキング方式）をあげていることから、広く利用されることが想定される DNS ブロッキング方式についての標準的な実施方法等をドキュメント化することが児童ポルノ対策を推進する上でも重要となってきている。

本ガイドラインでは、DNS ブロッキング方式について具体的な設定方法や導入において注意すべき点について運用面も含めて解説を行うものであり、DNS を利用したブロッキング導入に向けて ISP が検討を行う上での参考に資することを目的としている。

2. 児童ポルノ流通防止におけるブロッキングの位置づけ

ブロッキングは利用者がアクセスしようとするサイトのホスト名、IP アドレス、URL 等の情報を ISP が監視し、それがブロッキング対象であった場合に利用者の同意を得ることなくその通信を遮断する行為であるが、これは通信の秘密を侵害する行為である。しかし、児童の権利を著しく侵害し、児童からの性的搾取ないし性的虐待というべき児童ポルノ画像を掲載しているサイトに対するアクセスについては、そのサイトの検挙や削除が著しく困難な場合に限り、より侵害性の少ない手法と考えられるブロッキングを実施することでサイトへのアクセスを抑止することは許容されるものであると考えられる。[2]

3. DNS ブロッキング方式の概要

ブロッキングの方式には大きく分けて、①DNS により、ホスト名あるいはドメイン名を IP アドレスに変換する際にブロッキングを行う「DNS ブロッキング方式」、②本通信の際に IP ヘッダ内の宛先 IP アドレスもしくは HTTP コンテンツ部に含まれるアクセス先 URL 情報を元にブロッキ

[1] http://www.netsafety.or.jp/

[2] 詳細については法的問題検討 SWG 報告書参照のこと（http://good-net.jp/files/20110210114454.pdf）

ングを行う「パケットフィルタリング方式」、③HTTP プロキシにより HTTP 通信を一旦終端した上でアクセス先 URL 情報を元にブロッキングを行う「プロキシ方式」、④これらの方式の組合せによりブロッキングを行う「ハイブリッドフィルタリング方式」の４つの方式に分類することができる。[3]

このうちの DNS ブロッキング方式は、通信の際に行う DNS の名前解決の要求に対して、該当のドメインあるいはホスト名に対応する実際の IP アドレスを端末側に応答するのではなく、児童ポルノ掲載サイトへアクセスしようとしていることを警告するサイトの IP アドレスを代わりに応答することで、利用者が児童ポルノサイトに閲覧することをブロックする方式である。新たに大きな設備投資を行うことなく導入が容易であると考えられている一方で、ドメイン単位あるいはホスト名単位のブロッキングであるため、児童ポルノとは関係ないコンテンツまでブロックしてしまう「オーバーブロッキング」が発生することが問題であると考えられている。

DNS ブロッキングを実施するに際しては、児童ポルノ掲載サイトについて一律にドメイン部分を抽出してアドレスリストを作成するのではなく、ドメインあるいはホスト単位でのブロッキングが許容されると考えられる判定基準を策定し、DNS ブロッキング用のアドレスリストを作成することでオーバーブロッキングを極力回避するしくみとすることが重要である[4]。一方で、DNS ブロッキング方式は、画像単位でブロッキング可能な方式と比較するとブロッキング可能なサイトが少ないと考えられることから、効果は限定的である。ただし、導入が簡単なことから広く ISP として普及させることが容易な方式であると考えられ、最低限としての対策としては一定の効果は創出することができるものと考えられる。

[3] 各方式の詳細は、ISP 技術者サブワーキング報告書を参照のこと
(http://good-net.jp/usr/imgbox/pdf/20110411182350.pdf)
[4] アドレスリスト作成・管理の在り方サブワーキンググループ最終報告書を参照のこと
(http://)

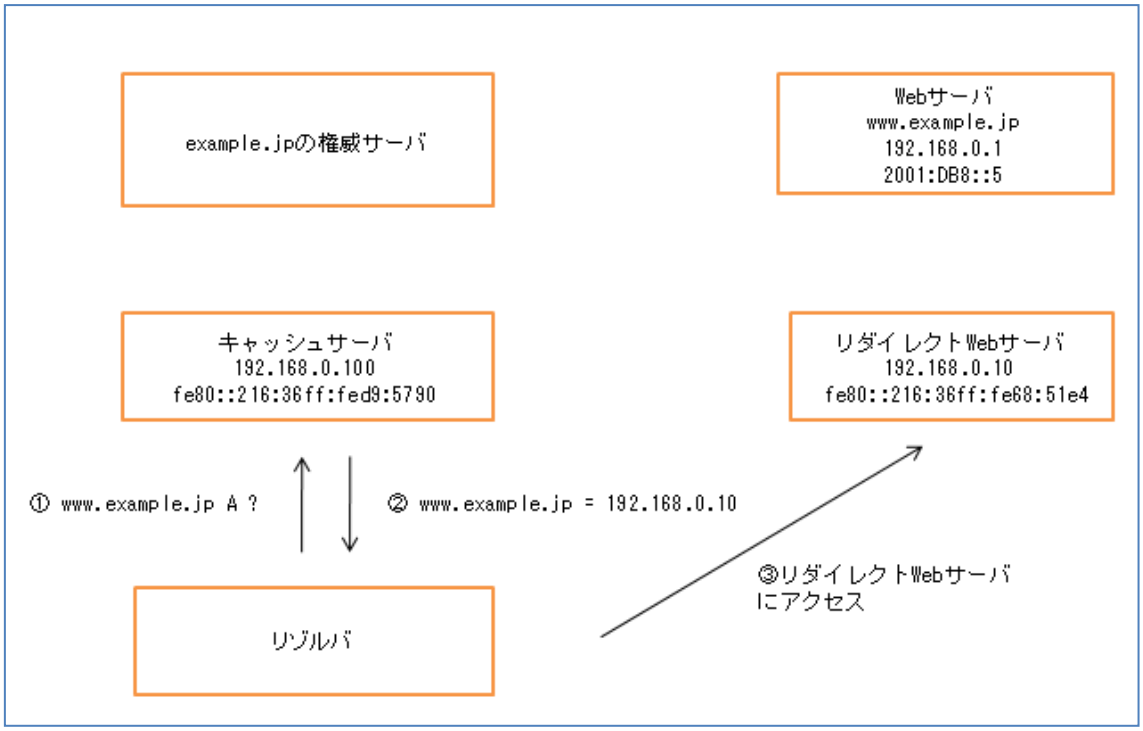
4. DNS ブロッキング方式の具体的な設定例

DNS によるブロッキングを実現する方法としては、ブロッキング対象のアドレスリスト（以下、ブロッキングアドレスリスト）をどこで管理するかによって、①DNS キャッシュサーバそれ自体でアドレスリストを保持・管理する方法、②DNS キャッシュサーバとは別サーバにてアドレスリストを保持・管理する方法の２つの方法が考えられる。ここでは、一般的に広く ISP にて利用されていると思われる Internet Systems Consortium の BIND[5]と NLnet Labs の unbound[6]の２つのオープンソースソフトによって、これらの２つの方法における具体的な設定例を紹介する。

4.1 キャッシュサーバ上でブロッキングリストを保持する方式(方式 1)の設定例

4.1.1 構成

example.jp ドメイン内の Web サイト www.example.jp (192.168.0.1)へのアクセスに対してブロッキングを行い、その通信をリダイレクト用 Web サーバ(192.168.0.10)に誘導し、そこでブロッキング警告画面を表示させる設定について説明する。図１にあるように、ISP にて運用中の DNS キャッシュサーバに加えて、ブロッキング警告画面として利用するリダイレクト用 Web サーバを準備する必要がある。



[5] http://www.isc.org/software/bind
[6] http://www.unbound.net/

図 1　DNS キャッシュサーバでブロッキングリストを保持する場合の構成例

4.1.2　DNS キャッシュサーバの設定[7]

4.1.2.1　BIND での設定例

ここでは、BIND9.7.2-P3 (2010 年 11 月 30 日リリース)での設定方法について説明する。

①　DNS キャッシュサーバ上にブロッキングする www.example.jp ゾーンを作成する。

ブロッキングする www.example.jp をマスタゾーンとして named.conf ファイルにて下記のように定義する。

named.conf

```
zone "www.example.jp" {
        type master;
        file "www.example.jp.db";
        };
```

②　www.example.jp のゾーンファイル www.example.jp.db を作成する。

www.example.jp の A、AAAA レコードとして、リダイレクトさせるリダイレクト用 Web サーバの IP アドレス 192.168.0.10 および fe80::216:36ff:fe68:51e4 を www.example.jp.db ファイルに登録する。

www.example.jp.db

```
$TTL 10
www.example.jp.    10      IN SOA  admin.www.example.jp.  admin.www.example.jp. (
                                   2010120908   ; serial
                                   7200         ; refresh (2 hours)
                                   3600         ; retry (1 hour)
                                   604800       ; expire (1 week)
                                   600          ; minimum (10 minutes)
                                   )
```

[7] BIND で特定のドメインへの DNS 問合せをブロッキングする機能をもつ Response Policy Zone が開発中であり BIND9.8 から実装される予定である（ベータ版が BIND9.8.0 b1 でリリース済）

```
        10      IN          NS      ns.www.example.jp.

ns.www.example.jp.      10      IN      A       192.168.0.100
ns.www.example.jp.      10      IN      AAAA        fe80::216:36ff:fed9:5790
www.example.jp.     10      IN      A       192.168.0.10
www.example.jp.     10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

4.1.2.2 unbound での設定例

unbound 1.4.7 (2010 年 11 月 8 日リリース)での設定方法について説明する。

DNS キャッシュサーバにおいて、local-data オプションを使用し、www.example.jp の A レコードおよび AAAA レコードとして、リダイレクト先となるリダイレクト用 Web サーバの IP アドレス 192.168.0.10 および fe80::216:36ff:fe68:51e4 を unbound.conf ファイルに登録する。

unbound.conf

```
local-data: "www.example.jp 10 IN A 192.168.0.10"
local-data: "www.example.jp 10 IN AAAA fe80::216:36ff:fe68:51e4"
```

4.1.3 リダイレクト用 Web サーバの設定

ここでは Apache2.2.14 を使用して、リダイレクト用 Web サーバの設定方法について説明する。リダイレクト用 Web サーバにおいては、実際にアクセスしようとするドメイン部分がこの Web サーバの IP アドレスに変換されるが、HTTP ホストヘッダ、URL パスはそのまま引き継がれることを想定して、HTTP ホストヘッダ、URL パスがどのような文字列があってもブロッキング警告画面を表示することが必要となる。

① ブロッキングされた旨を警告する警告画面 index.html を準備する。[8]

index.htm

```
<html>
<body>
DNS ブロッキングにより、リダイレクトされました
</body>
</html>
```

[8] 実施のブロッキング警告画面についてはアドレスリスト管理団体から ISP に対して共通なものが提供される(7.3 項参照)

② ブロッキング対象アドレス(www.example.jp)に対するアクセスに対してブロッキング警告画面(www.redirect-example.jp/index.html)が表示されるように、httpd.conf ファイルにて、VirtualHost を 2 つ作成し、リダイレクト先の指定を行うことで 1 台のサーバにて設定を行うことが可能になる。

下記の httpd.conf の例において、1 番目の VirtualHost は、ブロッキング警告画面にリダイレクトするための設定で、RedirectMatch (.*)の記述により、このサイトにアクセスしようとする URL パスがどのような文字列でも、www.redirect-example.jp/index.html にリダイレクトされる。このリダイレクトに際しては HTTP ステータスコードとしては、307 Temporary Redirect を返すことにする。また、ServerName * により、HTTP ホストヘッダがどのような文字列でも(www.redirect-example.jp は除く)、1 番目の VirtualHost にヒットする。この設定により、HTTP ホストヘッダ、URL パスがどのような文字列でも、1 番目の VirtualHost にマッチし、ブロッキング警告画面(www.redirect-example.jp/index.html)にリダイレクトされる。

リダイレクト後の HTTP ホストヘッダは www.redirect-example.jp となり、2 番目の ServerName www.redirect-example.jp と文字列が一致するため、2 番目の VirtualHost にマッチし、警告画面(http://www.redirect-example.jp/index.html)が表示される。

httpd.conf

```
# ブロッキング警告画面へのリダイレクト用 VirtualHost
<VirtualHost 192.168.0.10:80>
RedirectMatch 307 (.*) http://www.redirect-example.jp/index.html  ←リダイレクト先
ServerName *
ErrorLog /var/log/httpd/bad_error_log
TransferLog /var/log/httpd/bad_access_log
</VirtualHost>

# ブロッキング警告画面表示用
<VirtualHost 192.168.0.10:80>
DocumentRoot /var/www/html/redirect
ServerName www.redirect-example.jp  ←警告画面ホストのドメイン名を指定する
ErrorLog /var/log/httpd/redirect_error_log
TransferLog /var/log/redirect_access_log
</VirtualHost>
```

4.1.4 動作確認

① DNS キャッシュサーバ上で、dig により www.example.jp に対する名前解決を実施すし、回答として、書き換えられた 192.168.0.10 (A レコード)および fe80::216:36ff:fe68:51e4 (AAAA レコード)が得られるかどうかを確認する。

[BIND を使用した場合の動作確認による表示例]

A レコード

```
キャッシュサーバ# dig @127.0.0.1 www.example.jp A

; <<>> DiG 9.7.2-P3 <<>> @127.0.0.1 www.example.jp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26252
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         10      IN      A       192.168.0.10

;; AUTHORITY SECTION:
www.example.jp.         10      IN      NS      ns.www.example.jp.
```

```
;; ADDITIONAL SECTION:
ns.www.example.jp.      10      IN      A       192.168.0.100
ns.www.example.jp.      10      IN      AAAA    fe80::216:36ff:fed9:5790
```

AAAA レコード

```
root@ubuntu-4:~# dig @::1 www.example.jp AAAA

; <<>> DiG 9.7.2-P3 <<>> @::1 www.example.jp AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28992
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         10      IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
www.example.jp.         10      IN      NS      ns.www.example.jp.

;; ADDITIONAL SECTION:
ns.www.example.jp.      10      IN      A       192.168.0.100
ns.www.example.jp.      10      IN      AAAA    fe80::216:36ff:fed9:5790
```

② 同様に、リゾルバが DNS キャッシュサーバに www.example.jp の名前解決を依頼すると、その回答として書き換えられた 192.168.0.10 (A レコード)および fe80::216:36ff:fe68:51e4 (AAAA レコード)を得られることを確認する。

[BIND を使用した場合の動作確認による表示例]

A レコード

```
リゾルバ# dig @192.168.0.100 www.example.jp A
```

```
; <<>> DiG 9.7.0-P1 <<>> @192.168.0.100 www.example.jp A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55703
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         10      IN      A       192.168.0.10

;; AUTHORITY SECTION:
www.example.jp.         10      IN      NS      ns.www.example.jp.

;; ADDITIONAL SECTION:
ns.www.example.jp.      10      IN      A       192.168.0.100
ns.www.example.jp.      10      IN      AAAA    fe80::216:36ff:fed9:5790
```

AAAA レコード

```
リゾルバ# dig @fe80::216:36ff:fed9:5790 www.example.jp AAAA

; <<>> DiG 9.7.0-P1 <<>> @fe80::216:36ff:fed9:5790 www.example.jp AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10709
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

```
;; AUTHORITY SECTION:
www.example.jp.          10      IN      NS      ns.www.example.jp.

;; ADDITIONAL SECTION:
ns.www.example.jp.       10      IN      A       192.168.0.100
ns.www.example.jp.       10      IN      AAAA    fe80::216:36ff:fed9:5790
```

③　リゾルバ上の Web ブラウザで www.example.jp へアクセスしようとした際に、本来アクセスしようとしたサイトではなくリダイレクト用 Web サーバへアクセスされ、ブロッキング警告画面が表示されることを確認する。この際にはリゾルバは DNS キャッシュサーバとして 192.168.0.100 または fe80::216:36ff:fed9:5790 を設定しているものとする。

4.1.5　DNS ブロッキング設定により、回答が書き換えられる影響範囲

4.1.2 項での設定のように、www.example.jp がアドレスリストに掲載されている場合の設定を DNS キャッシュサーバに行うことで、ブロッキング対象としてブロッキングの設定がされたドメインあるいはホスト名（この例の場合は、www.example.jp）については表１および表２のように DNS のリソースレコードが書き換えられる。www.example.jp 以外の example.jp ドメインのサブドメインあるいはホスト名については、それがブロッキング対象のものと同一ドメイン(example.jp)であったとしても、完全に一致しない場合には該当ドメインの正規な権威サーバに対して DNS キャッシュサーバから問合せを行うことにより書き換えられていない正常な DNS の回答を返すことができる。ただし、example.jp ドメイン自体がアドレスリストに掲載され、example.jp ドメイン自体の設定が行われた場合、かつ、BIND を利用する場合においては、example.jp ドメインのサブドメインあるいはホスト名に対する名前解決については NXDOMAIN が返され名前解決に失敗することとなるため注意が必要である。unbound を利用する場合にはこの

ようなことは発生しない。

クエリ	クエリタイプ	問い合わせ先	名前解決結果
*.example.jp (www.example.jpは除く)	全てのクエリタイプ	example.jpの権威サーバ	example.jpの権威サーバからの回答
www.example.jp	SOA	キャッシュサーバ上のゾーンファイル	キャッシュサーバ上のSOA
	NS		キャッシュサーバ上のNS
	A		キャッシュサーバ上のA
	AAAA		キャッシュサーバ上のAAAA
	その他		登録されていないレコードの回答は得られない

（注：＊ は任意の文字列）

表１ DNS キャッシュサーバが BIND の場合の名前解決結果

クエリ	クエリタイプ	問い合わせ先	名前解決結果
*.example.jp (www.example.jpは除く)	全てのクエリタイプ	example.jpの権威サーバ	example.jpの権威サーバからの回答
www.example.jp	A	unbound.confのlocal-dataの情報	local-data A の情報
	AAAA		local-data AAAA の情報
	その他		local-dataに登録されていないレコードの回答は得られない

（注：＊ は任意の文字列）

表２ DNS キャッシュサーバが unbound の場合の名前解決結果

4.2 別サーバにてブロッキングリストを保持する方式（方式 2）の設定例

4.2.1 構成

図 2 にあるように、ブロッキングの対象となるドメインについてのゾーンファイルを一元的に管理するブロッキングリスト管理 DNS サーバを DNS キャッシュサーバとは別に用意し、DNS キャッシュサーバはブロッキングアドレスリスト対象のドメインあるいはホスト名に対する DNS 問合せについてはブロッキングリスト管理 DNS サーバへその問合せを転送する。ブロッキングリスト管理 DNS サーバでは、問合せに対してリダイレクト先 Web サーバの IP アドレスを回答することで閲覧者に対してブロッキング警告画面を表示させる。以下では、example.jp ゾーンの Web サイト www.example.jp (192.168.0.1)へのアクセスをブロックし、リダイレクト先 Web サーバ(192.168.0.10)に誘導する方法について説明する。

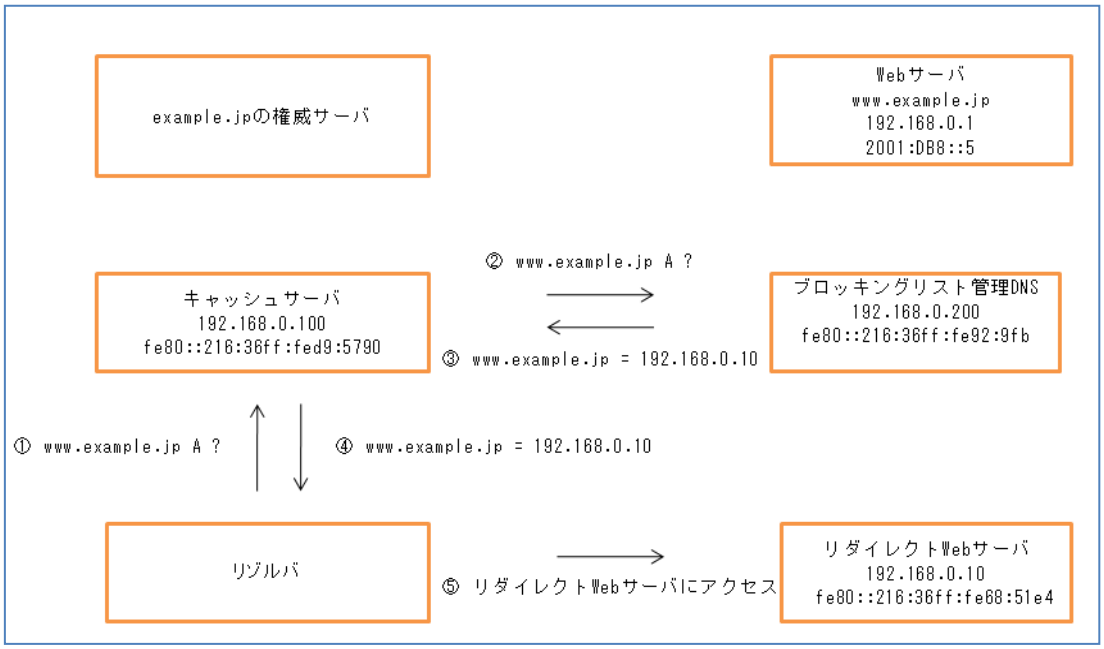


図２　**別サーバにてブロッキングリストを保持する方式場合の構成例**

4.2.2 DNS キャッシュサーバの設定例

4.2.2.1 BIND での設定例

ここでは BIND9.7.2-P3（2010 年 11 月 30 日リリース）を使用した設定例について説明する。

named.conf ファイルにおいて、forward オプションを使用し、ブロッキング対象である www.example.jp に関する DNS 問合せについてはブロッキングリスト管理 DNS サーバ（192.168.0.200 および fe80::216:36ff:fed9:5790）に転送させるよう設定する。その場合、forward only とすることで、ブロッキングリスト管理 DNS サーバのみに問合せを行うよう設定を行う。

name.conf

```
zone "www.example.jp" {
        type forward;
        forward only;
        forwarders {
                192.168.0.200;
                fe80::216:36ff:fe92:9fb;
                };
```

```
};
```

4.2.2.2 unbound での設定例

ここでは unbound 1.4.7 (2010 年 11 月 8 日リリース)を使用した設定例について説明する。

unbound.conf ファイルにおいて forward-zone オプションを使用し、ブロッキング対象である www.example.jp に関する DNS 問合せについてはブロッキングリスト管理 DNS サーバ(192.168.0.200 および fe80::216:36ff:fe92:9fb)に転送させるように設定を行う。

unbound.conf

```
forward-zone:
        name: "www.example.jp"
        forward-addr: 192.168.0.200
        forward-addr: fe80::216:36ff:fe92:9fb
```

4.2.3 ブロッキングリスト管理 DNS サーバの設定

4.2.3.1 BIND での設定例

ここでは BIND9.7.2-P3 (2010 年 11 月 30 日リリース)を使用した設定例について説明する。

① ブロッキング対象である www.example.jp をゾーンとして named.conf ファイルに登録し、www.example.jp をマスタゾーンとして定義する。

named.conf

```
zone "www.example.jp" {
        type master;
        file "www.example.jp.db";
        };
```

② www.example.jp ゾーンのゾーンファイル として www.example.jp.db ファイルを作成する。

www.example.jp.db ファイルでは www.example.jp の A レコードとして、リダイレクト先 Web サーバの IP アドレス 192.168.0.10 および fe80::216:36ff:fe68:51e4 を登録する。

www.example.jp.db

```
$TTL 10
www.example.jp.    10      IN SOA  admin.www.example.jp.  admin.www.example.jp.  (
                                        2010120908    ; serial
                                        7200          ; refresh (2 hours)
                                        3600          ; retry (1 hour)
                                        604800        ; expire (1 week)
                                        600           ; minimum (10 minutes)
                                        )
        10      IN          NS      ns.www.example.jp.

ns.www.example.jp.      10      IN      A       192.168.0.200
ns.www.example.jp.      10      IN      AAAA    fe80::216:36ff:fe92:9fb
www.example.jp.         10      IN      A       192.168.0.10
www.example.jp.         10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

4.2.3.2 unbound での設定例

ここでは unbound 1.4.7 (2010 年 11 月 8 日リリース)を使用した設定例について説明する。

unbound.conf ファイルにおいて local-data オプションを使用し、ブロッキング対象である www.example.jp の A レコードおよび AAAA レコードとしてリダイレクト先 Web サーバの IP アドレス 192.168.0.10 および fe80::216:36ff:fe68:51e4 を登録する。

unbound.conf

```
local-data: "www.example.jp 10 IN A 192.168.0.10"
local-data: "www.example.jp 10 IN AAAA fe80::216:36ff:fe68:51e4"
```

4.2.4 リダイレクト用 Web サーバの設定

方式 1 の場合と同様の手順により設定を行う（4.1.3 項参照）。

4.2.5 動作確認

① ブロッキングリスト管理 DNS サーバが正常にブロッキングドメインを読み込んでいるか確認するため、ブロッキングリスト管理 DNS サーバ上で www.example.jp の名前解決を行い、それに対する回答として Answer Section に書き換えられた回答である IP アドレス 192.168.0.10(A レコ

ード)、および、fe80::216:36ff:fe68:51e4(AAAA レコード)が得られることを確認する。

[BIND を使用した場合の動作確認による表示例]

A レコード

```
ブロッキングリスト管理 DNS# dig @127.0.0.1 www.example.jp A

; <<>> DiG 9.7.2-P3 <<>> @127.0.0.1 www.example.jp A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45864
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         10      IN      A       192.168.0.10
```

AAAA レコード

```
ブロッキングリスト管理 DNS# dig @::1 www.example.jp AAAA

root@ubuntu-4:~# dig @::1 www.example.jp AAAA

; <<>> DiG 9.7.2-P3 <<>> @::1 www.example.jp AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55955
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                    IN      AAAA
```

```
;; ANSWER SECTION:
www.example.jp.          10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

② 次に、DNS キャッシュサーバ上で www.example.jp の名前解決を実施することで、それに対する回答として書き換えられた回答である 192.168.0.10(A レコード)、および、fe80::216:36ff:fe68:51e4(AAAA レコード)が得られることを確認する。

[BIND を使用した場合の動作確認による表示例]

A レコード

```
キャッシュサーバ# dig @127.0.0.1 www.example.jp A

; <<>> DiG 9.7.2-P3 <<>> @127.0.0.1 www.example.jp A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43660
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.          10      IN      A       192.168.0.10
```

AAAA レコード

```
root@ubuntu-4:~# dig @::1 www.example.jp AAAA

; <<>> DiG 9.7.2-P3 <<>> @::1 www.example.jp AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36821
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
```

```
;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

③　同様に、リゾルバが DNS キャッシュサーバ(192.168.0.100)に www.example.jp の名前解決を実施することで、それに対する回答として書き換えられた回答 192.168.0.10(A レコード)および fe80::216:36ff:fe68:51e4(AAAA レコード)が得られることを確認する。

[BIND を使用した場合の動作確認による表示例]

A レコード

```
リゾルバ# dig @192.168.0.100 www.example.jp A

; <<>> DiG 9.7.0-P1 <<>> @192.168.0.100 www.example.jp A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60923
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         10      IN      A       192.168.0.10
```

AAAA レコード

```
リゾルバ# dig @fe80::216:36ff:fed9:5790 www.example.jp AAAA

; <<>> DiG 9.7.0-P1 <<>> @fe80::216:36ff:fed9:5790 www.example.jp AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28214
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         10      IN      AAAA    fe80::216:36ff:fe68:51e4
```

④　リゾルバ上の Web ブラウザからブロッキング対象サイト www.example.jp にアクセスすると、リダイレクト先 Web サーバでブロッキング警告画面が表示されるかを確認する。
リゾルバにおいて DNS キャッシュサーバは 192.168.0.100 に設定しているものとする。

4.2.6 DNS ブロッキング設定により、回答が書き換えられる影響範囲

4.1.5 と同様、www.example.jp を DNS キャッシュサーバに設定した場合には、ブロッキング対象ドメインあるいはホスト名（この例の場合は、www.example.jp）に完全一致した場合にのみ DNS 問合せの回答が書き換えられ、www.example.jp 以外の example.jp ドメインのサブドメインやホスト名についてはブロッキング対象のドメイン名(example.jp)が含まれている場合においても、DNS 問合せは正規な権威サーバに対して DNS キャッシュサーバから問合せが行われることで、書き換えられていない正常な DNS の回答を返すことができる。ただし、example.jp ドメイン自体がアドレスリストに掲載され、example.jp ドメイン自体を DNS キャッシュサーバに設定した場合、かつ、BIND を利用する場合においては、example.jp ドメインのサブドメインおよびホスト名に対する名前解決については NXDOMAIN が返され名前解決に失敗することとなるため注意が必要である。unbound を利用する場合には、このようなことは発生しない。

クエリ	クエリタイプ	問い合わせ先	名前解決結果
*.example.jp (www.example.jpは除く)	全てのクエリタイプ	example.jpの権威サーバ	example.jpの権威サーバからの回答
www.example.jp	SOA	forwardで指定したDNS	forwardで指定したDNSのSOA
	NS		forwardで指定したDNSのNS
	A		forwardで指定したDNSのA
	AAAA		forwardで指定したDNSのAAAA
	その他		登録されていないレコードの回答は得られない

(注：＊は任意の文字列)

表 3 DNS キャッシュサーバが BIND の場合の名前解決結果

クエリ	クエリタイプ	問い合わせ先	名前解決結果
*.example.jp (www.example.jpは除く)	全てのクエリタイプ	example.jpの権威サーバ	example.jpの権威サーバからの回答
www.example.jp	A	forwardで指定したDNS	forwardで指定したDNSのlocal-data A
	AAAA		forwardで指定したDNSのlocal-data AAAA
	その他		登録されていないレコードの回答は得られない

(注：＊は任意の文字列)

表 4 DNS キャッシュサーバが unbound の場合の名前解決結果

4.3 方式比較および考察

これまでに設定方法について述べてきた２つの方式、ブロッキングアドレスリストを DNS キャッシュサーバにて保持する方式と別なドメインリスト管理サーバにて保持する方式について比較を行うと、後者はブロッキング対象ドメインのゾーンファイル自体は集中管理が可能ではあるが、設定作業に際して DNS キャッシュサーバ側においてもドメインリスト管理サーバへの DNS 問合せの転送設定がリスト追加に際して必要であること、およびリスト更新に際しては DNS キャッシュサーバ側においてもキャッシュクリア作業も必要となることからリストの集中管理による運用管理上のメリットはそれほど大きくないと考えられる。また、詳細は 5.2 項において詳しく述べるが、ブロッキング対象ドメインが DNSSEC 対応となった場合、ドメインを管理しているサーバにおいて鍵の生成を行わないと該当ドメインへの問合せが ServFail により名前解決ができなくなることから、ドメインリストを別のドメインリスト管理サーバに保持した場合は DNSSEC の鍵生成作業がドメインリスト管理サーバにおいて必要となってくる。これらのことを考えると、ブロッキングアドレスリストを DNS キャッシュサーバにて保持する方式の方が運用的な面からは望ましいと考えられる。

	DNS キャッシュサーバ上にリストを保持 （方式１）	DNS キャッシュサーバとは別サーバ上にリストを保持 （方式２）
ブロッキングリストを保持するサーバ	DNS キャッシュサーバ	DNS キャッシュサーバ ブロッキングリスト管理サーバ
ブロッキングリストの更新対象サーバ	DNS キャッシュサーバのリストを更新	DNS キャッシュサーおよびブロッキングリスト管理サーバでのリストを更新
ブロッキングリスト更新時のキャッシュクリア	必要なし キャッシュされているブロッキング対象ドメインの情報はリスト更新を実施することでリスト更新情報が DNS キャッシュサーバに反映される。	必要あり キャッシュされているブロッキング対象ドメインの情報は、DNS キャッシュサーバのクリアがされない限りそれが expire するまで ブロッキングリスト管理サーバの更新された情報への問合せを行わない。
DNSSEC の干渉	ブロッキング対象ドメインへの DNS 問合せの回答は BIND、unbound ともに DNSSEC 対応ではない回答がリゾルバに対して返される。	ブロッキング対象ドメイン用について署名するための秘密鍵および公開鍵をブロッキングリスト管理サーバにて作成する必要がある。この鍵生成を行わない場合、該当ドメインへの問合せは ServFail エラーがリゾルバに対して返され、ブロッキング警告画面の表示ができない。

表５　方式比較

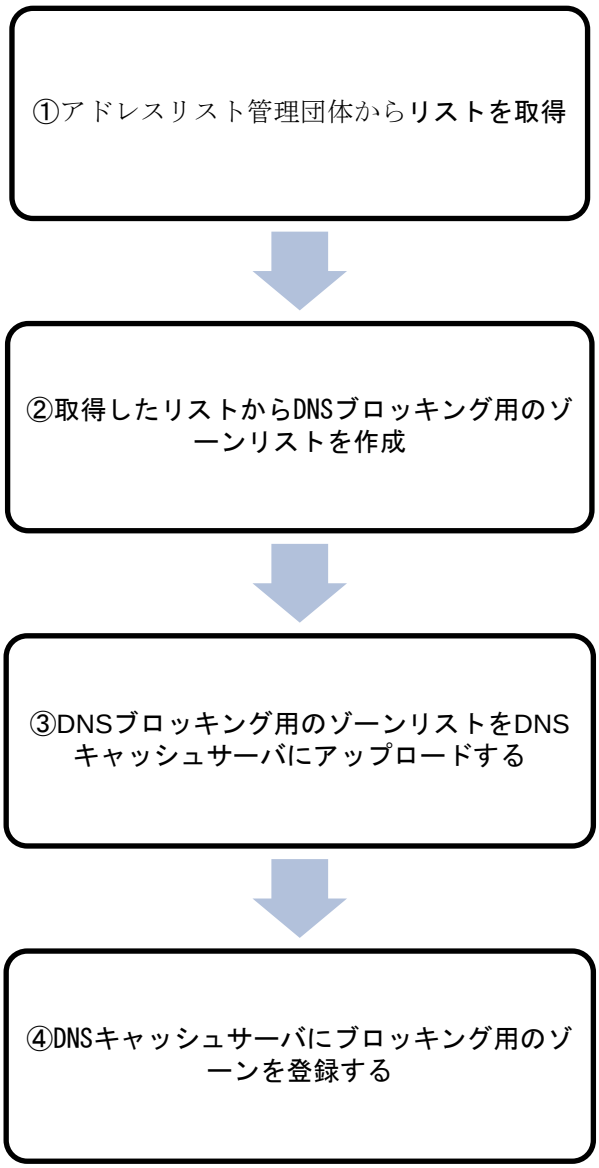
4.4. 導入手順

本項では、DNS ブロッキングの設定を行うためにいくつかのサンプルスクリプトを用いて、具体的なブロッキングを導入するための DNS キャッシュサーバの設定を説明する。

4.4.1 導入の全体の流れ

ブロッキングアドレスリストをアドレスリスト管理団体から取得し､そのリストからブロッキ

ング対象ドメインを抽出したリストについて DNS キャッシュサーバに設定を行い、ブロッキング対象ドメインを DNS キャッシュサーバに読み込む。この一連の手順は以下のような流れになる。



4.4.2 具体的な設定例

以下に、上記の手順に従って具体的な設定例の説明を行う。

① アドレスリスト管理団体からリストを取得

アドレスリスト管理団体からブロッキングアドレスリストを取得する。リストの受渡方法は7.2 項を参照されたい。取得したリストは、セキュリティ上、セキュアな領域に保存、アクセス制限を行うことが望ましい。

② 取得したリストからDNSブロッキング用のゾーンリストを作成

DNSブロッキング用ゾーンリストの作成について、awk スクリプトを使用した例を用いて説明を行う。awk スクリプトを実行できるマシン上にアドレスリスト管理団体より取得したリストがあるとして、DNSブロッキング用ゾーンリストの作成方法について説明する。

ここでは取得したリスト(CSV ファイル)の2列目 を”掲載ページのホスト名”、5列目を”掲載ページのブロッキング可否”として、それらの項目を抽出する例について記述する。

下記のawkスクリプトによって、DNSブロッキング用のリスト作成で必要となる2列目と5列目のみを抽出したときの表示例である。(1はブロッキング可、0はブロッキング否を意味する)

```
# awk -F, '{print $2,$5}' blocking_list_sample.csv
2 5
掲載ページのホスト名 掲載ページのDNSブロッキング可否
bad.example1.jp 1  ←ブロッキング可
abc.example1.jp 0  ←ブロッキング否
bad.example2.jp 1
bad.example3.jp 1
bad.example4.jp 1
abc.example2.jp 0
bad.example5.jp 1
bad.example6.jp 1
abc.example3.jp 0
bad.example7.jp 1
bad.example8.jp 1
bad.example9.jp 1
bad.example7.jp 1
bad.example8.jp 1
bad.example9.jp 1
bad.example10.jp 1
```

awk スクリプトで DNS ブロッキング可のホスト名のみ抽出したリスト blocking_list_sample.txt を作成する。このスクリプトは、リストの5列目(掲載ページのブロッキング可否)のフラグが1(DNSブロッキング可)のホスト名を抽出する。

```
# awk -F, '$5==1{print $2}' blocking_list_sample.csv  > blocking_list_sample.txt
```

awk スクリプトを実行すると、下記のリスト(blocking.txt)が生成される。

```
# cat blocking_list_sample.txt
bad.example1.jp
bad.example2.jp
bad.example3.jp
bad.example4.jp
bad.example5.jp
bad.example6.jp
bad.example7.jp
bad.example8.jp
bad.example9.jp
bad.example10.jp
```

③　DNS ブロッキング用のゾーンリストを DNS キャッシュサーバにアップロードする

上記②で作成した blocking_list_sample.txt を DNS キャッシュサーバにアップロードする。セキュリティ上、SFTP、SCP などセキュアな通信でアップロードすることが望ましい。

④　DNS キャッシュサーバにブロッキング用のゾーンを登録する

[BIND を利用する場合のゾーンの登録の設定例]

次の２つのサンプルスクリプトを用いて、具体的にブロッキング用ゾーンファイルおよび named.conf の作成を具体的に実施する。

まず、設定用ファイルとして、以下の３つのファイルを準備する。

・ blocking_list_sample.txt
　上記②で作成したブロッキング対象ドメインをリストとして記述したファイル

・ template_zone.txt
　ブロッキング対象ドメインのゾーンファイルのテンプレートファイル。テンプレートファイルの中では、リダイレクト先 Web サーバの IP アドレスや DNS キャッシュサーバの IP アドレスについてはあらかじめ記入しておく。

```
# less -N template_zone.txt
     1 $TTL    10
     2 @       IN      SOA     DOMAIN. root.DOMAIN.  (
     3                                  2011011701      ; Serial
     4                                  3600            ; Refresh
     5                                  900             ; Retry
     6                                  3600000         ; Expire
     7                                  3600 )          ; Minimum
     8         IN      NS      ns1.DOMAIN.
     9         IN      NS      ns2.DOMAIN.
    10         IN      A       192.168.0.1 ←リダイレクト先 Web サーバの IP
    11 ns1     IN      A       10.0.0.1 ←DNS キャッシュサーバの IP
    12 ns2     IN      A       10.0.0.2 ←DNS キャッシュサーバの IP
```

・ template_named.conf

named.conf のテンプレートファイル。ここで、文字列 DOMAIN は、ブロッキングリスト blocking_list_sample.txt に記載されているドメインに置換される。

```
# less -N template_named.conf.txt
     1 zone "DOMAIN" {
     2         type master;
     3         file "/var/named/blocking/DOMAIN.zone";
     4         notify no;
     5         allow-update { none; };
     6         }
```

・create_blocking_zone.sh

テンプレートゾーンを記述した template_zone.txt ファイルとブロッキング対象ドメインのリストである blocking_list_sample.txt ファイルからブロッキング用ドメインのゾーンファイルを作成するスクリプト

```
# less -N create_blocking_zone.sh
     1 #!/bin/bash
     2
     3 DATAFILE=blocking_list_sample.txt
     4 TEMPLATE=template_zone.txt
     5
     6 for data in $(cat $DATAFILE)
```

```
 7 do
 8          dom=${data%:*}
 9          sed "{ s/DOMAIN/$dom/g; }" $TEMPLATE > $dom.zone
10 done
```

・create_blocking_named.conf.sh

ブロッキング対象ドメインのリストである blocking_list_sample.txt ファイルと named.conf ファイルのテンプレートである template_named.conf.txt ファイルからブロッキングを実施する DNS キャッシュサーバの named.conf である blocking_zone_named.conf を作成するスクリプト

```
# less -N create_blocking_named.conf.sh
 1 #!/bin/bash
 2
 3 DATAFILE=blocking_list_sample.txt
 4 TEMPLATE=template_named.conf.txt
 5
 6 rm -f blocking_zone_named.conf
 7
 8 for data in $(cat $DATAFILE)
 9 do
10          dom=${data%:*}
11          sed "{ s/DOMAIN/$dom/g; }" $TEMPLATE >> blocking_zone_named.conf
12 done
```

(a) create_blocking_zone.sh を実行することにより、実行したディレクトリ上にブロッキング対象ドメインのゾーンファイルがドメイン名.zone というファイル名で生成される。

```
# ./create_blocking_zone.sh
```

```
# ls *.zone
bad.example1.jp.zone              bad.example3.jp.zone              bad.example6.jp.zone
bad.example9.jp.zone
bad.example10.jp.zone  bad.example4.jp.zone  bad.example7.jp.zone
bad.example2.jp.zone   bad.example5.jp.zone  bad.example8.jp.zone
```

また、ドメイン毎のゾーンファイルが下記のように生成される。

```
# cat bad.example1.jp.zone
$TTL    10
```

```
@       IN      SOA     bad.example1.jp.  root.bad.example1.jp.   (
                                2011011701      ; Serial
                                3600            ; Refresh
                                900             ; Retry
                                3600000         ; Expire
                                3600 )          ; Minimum
        IN      NS      ns1.bad.example1.jp.
        IN      NS      ns2.bad.example1.jp.
        IN      A       192.168.0.1
ns1     IN      A       10.0.0.1
ns2     IN      A       10.0.0.2
```

(b)　create_blocking_named.conf.sh を実行することにより、実行したディレクトリに ブロッキングを実施する DNS キャッシュサーバ用の named.conf である blocking_zone_named.conf ファイルが生成される。

```
# ./create_blocking_named.conf.sh
```

```
# cat blocking_zone_named.conf
zone "bad.example1.jp" {
        type master;
        file "/var/named/blocking/bad.example1.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example2.jp" {
        type master;
        file "/var/named/blocking/bad.example2.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example3.jp" {
        type master;
        file "/var/named/blocking/bad.example3.jp.zone";
        notify no;
```

```
        allow-update { none; };
        };

zone "bad.example4.jp" {
        type master;
        file "/var/named/blocking/bad.example4.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example5.jp" {
        type master;
        file "/var/named/blocking/bad.example5.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example6.jp" {
        type master;
        file "/var/named/blocking/bad.example6.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example7.jp" {
        type master;
        file "/var/named/blocking/bad.example7.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example8.jp" {
        type master;
        file "/var/named/blocking/bad.example8.jp.zone";
        notify no;
        allow-update { none; };
```

```
        };

zone "bad.example9.jp" {
        type master;
        file "/var/named/blocking/bad.example9.jp.zone";
        notify no;
        allow-update { none; };
        };

zone "bad.example10.jp" {
        type master;
        file "/var/named/blocking/bad.example10.jp.zone";
        notify no;
        allow-update { none; };
        };
```

(c)　ブロッキング用ゾーンファイル(bad.example1.jp.zone～bad.example10.jp.zone)を/var/named/blocking ディレクトリ配下にコピーする。

```
# mv bad.example*.jp.zone /var/named/blocking/

# ls /var/named/blocking/
bad.example1.jp.zone    bad.example3.jp.zone   bad.example6.jp.zone
bad.example9.jp.zone   bad.example10.jp.zone   bad.example4.jp.zone
bad.example7.jp.zone   bad.example2.jp.zone    bad.example5.jp.zone
bad.example8.jp.zone
```

(d)　ブロッキング用の blocking_zone_named.conf を /etc ディレクトリ配下にコピーする。

```
# mv blocking_zone_named.conf /etc/
```

(e)　include オプションを使用し、blocking_zone_named.conf を読み込むように named.conf を編集する。

named.conf

```
# Add blocking zones as master
include "/etc/blocking_zone_named.conf";
```

(f) rndc reload でコンフィグレーションファイルのリロードを実施する。

```
# rndc reload
server reload successful
```

(g) syslog にブロッキング用のゾーンを読み込んだログが表示されることを確認する。

```
named[17097]: zone bad.example1.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example10.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example2.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example3.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example4.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example5.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example6.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example7.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example8.jp/IN: loaded serial 2011011701
named[17097]: zone bad.example9.jp/IN: loaded serial 2011011701
```

(h) 上記作業を各 DNS キャッシュサーバに対して実施する。

(i) ブロッキング対象ドメインに対し dig により名前解決を実施すると、リダイレクト先 Web サーバの IP アドレス 192.168.0.1 の応答が戻ってくることを確認する。

```
# dig @127.0.0.1 bad.example1.jp

; <<>> DiG 9.7.2-P3 <<>> @127.0.0.1 bad.example1.jp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64216
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;bad.example1.jp.                       IN      A

;; ANSWER SECTION:
bad.example1.jp.            10      IN      A       192.168.0.1
```

```
;; AUTHORITY SECTION:
bad.example1.jp.        10      IN      NS      ns1.bad.example1.jp.
bad.example1.jp.        10      IN      NS      ns2.bad.example1.jp.

;; ADDITIONAL SECTION:
ns1.bad.example1.jp.    10      IN      A       10.0.0.1
ns2.bad.example1.jp.    10      IN      A       10.0.0.2
```

[unbound を利用する場合のゾーンの登録の設定例]

ここでも以下の２つのサンプルスクリプトを用いて、具体的にブロッキング用ゾーンファイルおよび unbound.conf の作成を具体的に実施する。

・ create_blocking_unbound.conf.sh

ブロッキングアドレスリスト blocking_list_sample.txt から DNS キャッシュサーバにおけるブロッキング用のコンフィグレーションファイル blocking_unbound.conf を作成する。このスクリプトでは、12 行目の IP アドレスにリダイレクト先 Web サーバの IP アドレスを記述する。12 行目の文字列 $dom はブロッキングリスト blocking_list_sample.txt に記載されているドメインに置換される。

```
# less -N create_blocking_unbound.conf.sh
     1 #!/bin/bash
     2
     3 DATAFILE=blocking_list_sample.txt
     4
     5 rm -f blocking_unbound.conf
     6
     7 echo "server:" >> blocking_unbound.conf
     8
     9 for data in $(cat $DATAFILE)
    10 do
    11 dom=${data%:*}
    12 echo "local-data: ¥"$dom. 10 IN A 192.168.0.1¥"" >> blocking_unbound.conf
    13 done
```

(a) create_blocking_unbound.conf.sh を実行する。実行したディレクトリ上に

blocking_unbound.conf というファイルが生成される。

```
# ./create_blocking_unbound.conf.sh
```

```
# cat blocking_unbound.conf
server:
local-data: "bad.example1.jp. 10 IN A 192.168.0.1"
local-data: "bad.example2.jp. 10 IN A 192.168.0.1"
local-data: "bad.example3.jp. 10 IN A 192.168.0.1"
local-data: "bad.example4.jp. 10 IN A 192.168.0.1"
local-data: "bad.example5.jp. 10 IN A 192.168.0.1"
local-data: "bad.example6.jp. 10 IN A 192.168.0.1"
local-data: "bad.example7.jp. 10 IN A 192.168.0.1"
local-data: "bad.example8.jp. 10 IN A 192.168.0.1"
local-data: "bad.example9.jp. 10 IN A 192.168.0.1"
local-data: "bad.example10.jp. 10 IN A 192.168.0.1"
```

(b) include オプションを使用し、blocking_unbound.conf を読み込むように unbound.conf を編集する。

unbound.conf

```
# Add for blocking
include: "/usr/local/etc/unbound/blocking_unbound.conf"
```

(c) blocking_unbound.conf を include オプションで指定したディレクトリにコピーする。

```
# mv blocking_unbound.conf /usr/local/etc/unbound/
```

(d) unbound-control reload を実行しコンフィグレーションファイルをリロードする。

```
# unbound-control reload
ok
```

(e) unbound-control list_local_data コマンドで、local-data として読み込んでいるドメイン名がブロッキング対象ドメインであることを確認する。

```
# unbound-control list_local_data | grep bad
bad.example1.jp.    10      IN      A       192.168.0.1
bad.example10.jp.   10      IN      A       192.168.0.1
bad.example2.jp.    10      IN      A       192.168.0.1
bad.example3.jp.    10      IN      A       192.168.0.1
```

bad.example4.jp.	10	IN	A	192.168.0.1
bad.example5.jp.	10	IN	A	192.168.0.1
bad.example6.jp.	10	IN	A	192.168.0.1
bad.example7.jp.	10	IN	A	192.168.0.1
bad.example8.jp.	10	IN	A	192.168.0.1
bad.example9.jp.	10	IN	A	192.168.0.1

(f)　ブロッキングリストに表示されているドメインに対して dig コマンドを実施し、リダイレクト先 Web サーバの IP アドレス(192.168.0.1)が応答として返ってくることを確認する。

(g)　上記作業をキャッシュサーバごとに実施する。

5. DNS ブロッキング導入に際しての懸念事項

5.1 サービス提供へ与える影響

ブロッキングを導入するに際して最も懸念される点は、ブロッキングの導入が DNS のシステムリソースに対してどのように、どの程度の影響を与えるか、さらにはその影響により自社サービスのサービス品質への影響が発生するのかどうかがある。ブロッキングの導入による DNS のシステムリソースに対する影響を判断することで、サービス品質への影響を回避するために設備増設の対応を考慮しなければならないのか、設備増設が必要な場合はどれくらいの規模の投資が新たに必要なのかということを検討することが必要となる。

サービスへの影響を検討するに際しては、主に２つの観点からの影響を考慮する必要がある。１つは、ブロッキングに関する設定を DNS キャッシュサーバに行うことによる DNS のシステムリソースに対する影響、もう１つは、ブロッキングアドレスリストを更新する処理がシステムリソースに与える影響である。本項では、DNS キャッシュサーバに対して一定量の DNS 問合せクエリによる負荷をかけた状態にて、DNS キャッシュサーバに対してブロッキングの設定を行う前後でのシステムリソースの変化、およびブロッキングアドレスリストの更新処理を行った際のシステムリソースの変化、DNS キャッシュサーバへの問合せクエリに対する応答の変化について測定を行った。測定に際しては、以下のパラメータを変化させて性能測定を行った。

① 一定量の DNS 問合せクエリ数（500qps、1000qps）
② ブロッキングアドレスリスト数（500、1000、3000、5000）
③ DNS キャッシュサーバにおけるキャッシュヒット率（0%、50%、80%）
④ ハードウェアスペック(Intel Pentium4 2.4GHz/メモリ 1GB、Intel Xeon X5650 2.67GHz/メモリ 8GB)

また、利用する DNS ソフトウェアとしては、広く利用されているオープンソースである BIND9.7.2-P3 および unbound1.4.7 にて測定を行った。以下に、それぞれのソフトウェアを使用した場合においての性能評価の結果から観察できた特徴について説明する。

5.1.1 ブロッキング設定が DNS サービスに与える影響

DNS キャッシュサーバのマシンスペックやブロッキングアドレスリスト数、キャッシュヒット率、DNS 問合せクエリ数、使用ソフトウェアについてどれを変化させたとしても、ブロッキング設定の前後で DNS キャッシュサーバの CPU 使用率は変わることはなかった。メモリ使用率については、BIND を利用した場合においては、ブロッキングアドレスリスト数が増えるに応じてメモリ増加量も増え、アドレスリスト数が 1,000 で約 15MB、3,000 で約 30MB、10,000 で約 90MB の増

加量があった。また、unbound を利用した場合においては、BIND の場合と比較してメモリ増加量は少なく抑えられ、アドレスリスト数が 10,000 の場合においても約 15MB の増加になった。

全体のメモリ量から考えると、これらのメモリ増加量はシステムが動作する上では比較的小さい影響であることから、CPU およびメモリの使用量の増加量の観点からは、ブロッキング導入によるシステムへの影響は特に考慮するほどのことでもないと考えられる。

5.1.2 ブロッキングアドレスリスト更新処理が DNS サービスに与える影響

5.1.2.1 BIND の場合

ハードウェアスペックの低いマシン(CentOS 5.5、Intel Pentium4 2.4GHz、メモリ 1GB)においては、アドレスリスト数を 3,000 までにするとリストの読み込み時に DNS からの応答がなくなりサービス断状態となった。処理できるアドレスリスト数はキャッシュヒット率が増えるに従って多くなり、キャッシュヒット率が 0%においてはリスト数 300、キャッシュヒット率 50%においてはリスト数 500、キャッシュヒット率 80%においてはリスト数 1,000 でリスト更新の際に DNS 問合せクエリに対する応答がリスト更新を実施している約 4 秒間の間処理できなくなる場合が発生することが観察できた。この傾向は、DNS 問合せクエリ数が 500qps および 1,000qps どちらの場合においても同様な傾向が見られ、クエリ処理数にはあまり依存することなく性能劣化がみられた。

ハードウェアスペックの高いマシン(CentOS 5.5、Intel Xeon X5650 2.67GHz、メモリ 8GB)で同様な性能試験を実施すると、アドレスリスト数が 1,000 においても、リスト更新処理時においても DNS 問合せクエリの処理を欠落させることなく動作させることができ、低スペックマシンと比較して処理できるリスト数はかなり改善することがわかった。アドレスリスト数が 3,000 の場合には、リスト更新処理中に通常時と比較して CPU 使用率が約 3～4 倍に、アドレスリスト数が 5,000 の場合には CPU 使用率が約 5～6 倍に上昇し、DNS 問合せクエリに対して処理が欠落する場合が発生した。また、アドレスリスト数が 10,000 になると、リロード中に約 3 秒間無応答状態となった。これらの傾向はキャッシュヒット率および DNS 問合せクエリ数が 500qps と 1,000qps のどちらの場合もほぼ同様な傾向となった。

5.1.2.2 unbound の場合

unbound を利用した場合では、BIND を利用した場合と比べてハードウェアマシンのスペックに関わらずより多くのアドレスリスト数に対して、サービスへの影響を与えることなく処理が可能であった。ハードウェアスペックの低いマシン(CentOS 5.5、Intel Pentium4 2.4GHz、メモリ 1GB)においては、キャッシュヒット率が大きいほど処理可能なアドレスリスト数も大きくなり、キャッシュヒット率 0%においてアドレスリスト数 300、ヒット率 50%でアドレスリスト数 3,000、

ヒット率 80%でアドレスリスト数 10,000 までサービスに影響なく処理が可能であった。また、ハードウェアスペックの高いマシン(CentOS 5.5、Intel Xeon X5650 2.67GHz、メモリ 8GB)においては、キャッシュヒット率の値に関わらずアドレスリスト数 10,000 においてもサービスに影響なくリスト更新処理を行うことができた。

これらのことから、提供サービスへの影響を回避するためにはアドレスリスト数の増加に応じてシステムのハードウェアスペックを高性能なものに見直す、もしくは利用する DNS ソフトウェアを unbound に変更する等の対応を検討していく必要がある。

5.2 DNSSEC 導入による影響

DNSSEC (DNS SECurity extentions) は DNS への問合せに対する回答を偽装する攻撃に対して、DNS の応答に署名情報を付加することで DNS の応答が正当であることを検証するしくみである。2010 年 7 月からドメインネームシステムの最上位のゾーンであるルートゾーンへの DNSSEC 導入が開始され、jp ゾーンにおいても DNSSEC 署名が 2010 年 10 月 17 日より開始されている。2011 年 1 月 16 日からは jp ドメイン名サービスへの署名鍵の登録受付を JPRS が開始しており[9]、今後一般的に広くドメインの DNSSEC 対応が進んでいくことが想定される。

DNS によるブロッキング方式は正当な DNS 応答をブロッキングのために別なものに書き換えることを行うことから、DNSSEC とブロッキングが併存した場合の影響を把握しておくことは非常に重要である。ここでは DNSSEC 導入による DNS キャッシュサーバ、リゾルバへの影響を説明する。4 項で述べた 2 つの方式、DNS キャッシュサーバ上でブロッキングリストを保持する方式(方式 1)と、別サーバにてブロッキングリストを保持する方式(方式 2)について、DNSSEC とブロッキングが併存した場合にどのような影響があるかを説明する。[10]

5.2.1 キャッシュサーバ上でブロッキングリストを保持する方式（方式１）の場合

DNSSEC は DNS クエリに対して外部から受け取った DNS 回答の妥当性、正当性を検証する仕組みであり、DNSSEC の検証は、DNS キャッシュサーバおよびリゾルバにおいて行われる。DNS キャッシュサーバは権威サーバからの DNS 回答を検証し、リゾルバは DNS キャッシュサーバからの回答を検証することとなる。

DNS キャッシュサーバ上にブロッキングリストをマスターゾーンとして保持しているばあにおいては、該当ドメインに関しての権威ゾーンとして動作するため、DNS キャッシュサーバにおいては DNSSEC の検証は行われない。そのため、この DNS キャッシュサーバからの回答は、ブ

9 プレスリリース「JPRS が JP ドメイン名サービスに DNSSEC を導入」(http://jprs.co.jp/press/2011/110117.html)

10 JPRS においても本件の考察がなされている。「DNS ブロッキングと DNSSEC を共存させるための手法について」(http://jprs.jp/tech/notice/2010-07-28-dns-blocking-dnssec.html)

ロッキングリストにあるドメインが DNSSEC 対応していたとしても本来の権威サーバへの問合せを行うことなく DNS キャッシュサーバより直接 DNSSEC 対応ではない回答を行うこととなり、その回答内容に従って該当通信はブロッキング警告画面の Web サーバにリダイレクトさせることができる。ただし、リゾルバ自身が DNSSEC による名前検証を実証する場合には、リゾルバや利用するアプリケーションの実装によっては DNSSEC 対応でない回答を受けることでリゾルバや利用するアプリケーションが正常に動作することができず利用者が影響を受ける可能性がある。

5.2.2 別サーバにてブロッキングリストを保持する方式（方式２）の場合

この方式では､ブロッキングリスト管理サーバがブロッキング対象ホストの権威サーバとなり、DNS キャッシュサーバはブロッキングリスト管理サーバから受け取った DNS 回答に対して DNSSEC の検証を行う。キャッシュサーバは DNSSEC の検証を実施した際、ルートサーバの鍵を使用した検証を行い、それが本来の情報から書き換えられた回答となっているため、検証失敗となる。その結果、DNS キャッシュサーバはリゾルバに対して DNS エラー(ServFail)を返すため、ブロッキング警告画面のある Web サイトに該当通信をリダイレクトすることができない。

この通信不可事象を回避する方法としては、該当ドメインに対する名前解決を DNSSEC での検証の対象外とする方法がある。BIND においては現状ではこのような設定をすることができないが、unbound においては domain-insecure オプションを利用することで設定が可能である。

下記に unbound.conf の設定例を示す。

unbound.conf

```
# ブロッキングする www.example.jp は DNSSEC の検証を実施しない
         domain-insecure: "www.example.jp"

forward-zone:
        name: "www.example.jp"
        forward-addr: 192.168.0.200

# ルートゾーンの鍵
        trust-anchor:           ".          DNSKEY          257          3          5
AwEAAcXQXclcC0EAHjGmYCqr0ppFUL/1XET/U+4Z7EJBEIiBr1SktS1y
GGEn5RPsW3+M2HvN/tCd0lJYB9CEVukBhsgpXjadBrGt4U24U80rK11V
aNG3zMmvGDYSUn4P7k+HbGHmaoF3qZE7ywtRu7HFR5B4MrldIECUDu0n
vQNCMt1jDPPJmnPOzBOTF4ZFh4xwjeN3SVuhHY6qGRu8WQ0EzebQFqkP
```

```
if0VEt1eUkbEWvePVnnsomfQEMSyi5Z00qP36/Z0+zj1o31Q4n65jS4P
yVbCaKnfSZVnb+WgUJYeHYP2E/EBZV3713Ij0MRIVFAAmkA4+grJTbra LPStMsqafXU=″
```

この例では、ルートゾーンの鍵の設定を行い DNS キャッシュサーバとしては DNSSEC の検証を行う設定をしているが、domain-insecure オプションを設定することで該当ドメインについては DNSSEC 検証の対象外となる。この設定での実際の動作について見てみると、

```
キャッシュサーバ# dig @127.0.0.1 www.example.jp +dnssec +multiline

; <<>> DiG 9.7.2-P3 <<>> @127.0.0.1 www.example.jp +dnssec +multiline
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8305
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.example.jp.                 IN A

;; ANSWER SECTION:
www.example.jp.          10 IN A 192.168.0.10
```

リゾルバから DNS キャッシュサーバに対して DO ビットを ON にした www.example.jp に関するの DNS クエリに対して、DNS 回答は DNSSEC の署名のない回答としてブロッキング警告画面の IP アドレスが回答される。

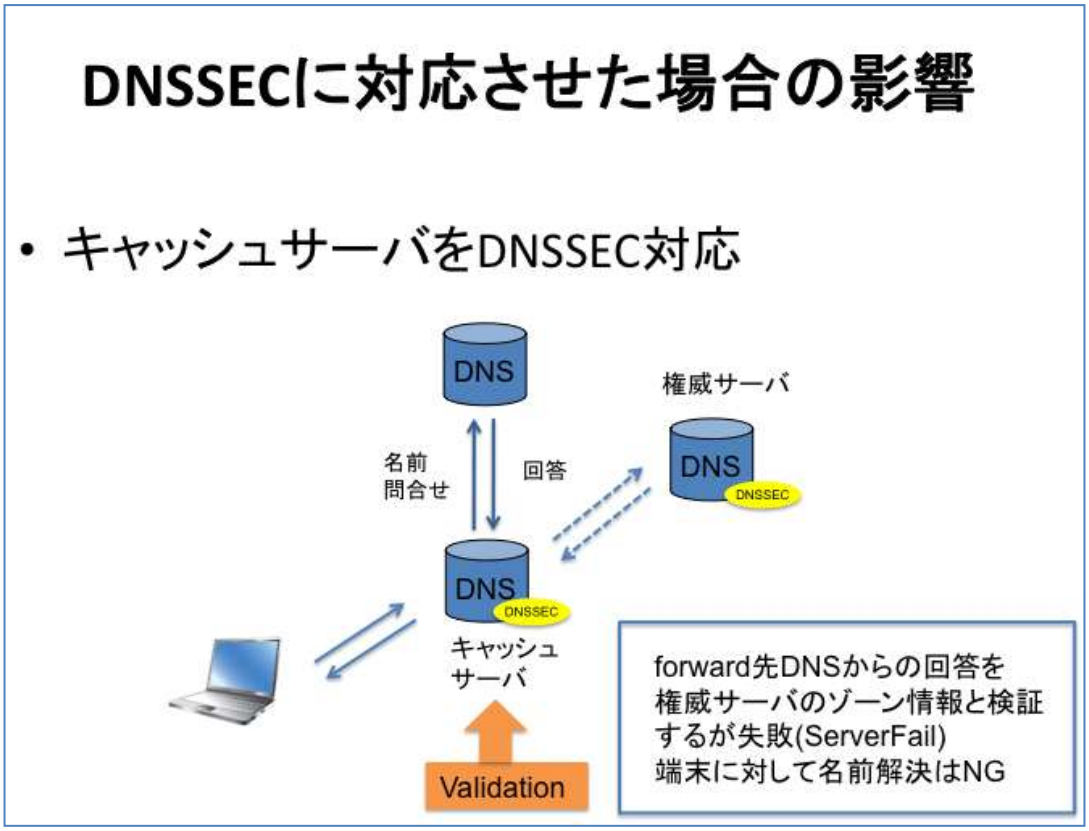


図５　DNSSEC に対応させた場合の影響

BIND においては、特定のドメインを DNSSEC 対象からはずすこのような設定オプションがないため、通信ができない事象を回避する方法としては、ブロッキングリスト管理サーバ側の該当ドメインのゾーンについて DNSSEC の署名を作成し、それを DNS キャッシュサーバに登録することで DNS キャッシュサーバでの DNSSEC 検証を成功させることは可能ではある。この場合は、問合せを受けたリゾルバに対しては DNSSEC の署名付きの回答として DNS 回答は行われるが正当な回答とは違う偽の署名付きの回答をすることになる。そのため、リゾルバにおいて DNS キャッシュサーバからの回答の検証を行った場合には DNSSEC の検証が失敗することになるため、この方法により完全に通信ができない事象を回避する方法とはならないことに気を付ける必要がある。

6. BIND Response Policy Zone (RPZ) を使用したDNSブロッキング方式

BIND 9.8以降に実装された Response Policy Zone (以降RPZ)を使用した DNSブロッキングについて説明する。

6.1 RPZ概要説明

RPZは、設定されたルールに基づき、DNS応答の書き換え、リゾルバへDNS応答する機能である。BIND 9.8以降では前述した、方式1、方式2に加え、RPZを使用することでも、DNSによるブロッキングが可能である。

RPZを実装したリスト配信サーバ(BIND)を用意し、配信サーバ上でブロッキングリストの一元管理、リスト配信サーバから複数台のキャッシュサーバへリストの一括配信が可能である。

なお、RPZはBIND9.8以降に実装された機能で、RPZによるDNSブロッキングを実現するためにはBIND9.8以降のバージョンを使用する必要がある。

なお、今回の検証では、RPZ使用時のパフォーマンス測定は実施していないので、RPZを使用したDNSブロッキングを導入する際は、導入前にパフォーマンス測定、サイジング設計を実施されたい。

6.2 RPZ構成例

RPZを使用したDNSブロッキングの構成例を2つ説明する。

6.2.1 RPZに対応したキャッシュサーバ上でブロッキンリストの更新を行う方式

example.jpドメイン内のWebサイト www.example.jp (192.168.0.1)へのアクセスに対してブロッキングを行い、その通信をリダイレクト用Webサーバ(192.168.0.10)に誘導し、そこでブロッキング警告画面を表示させる設定について説明する。

図6にあるように、ISPにて運用中のキャッシュサーバをRPZに対応させ、DNSブロッキングを行う。ブロッキングリストの更新はRPZを実装したキャッシュサーバ上で行う。

また、ブロッキング警告画面として利用するリダイレクト用Webサーバを準備する必要がある。

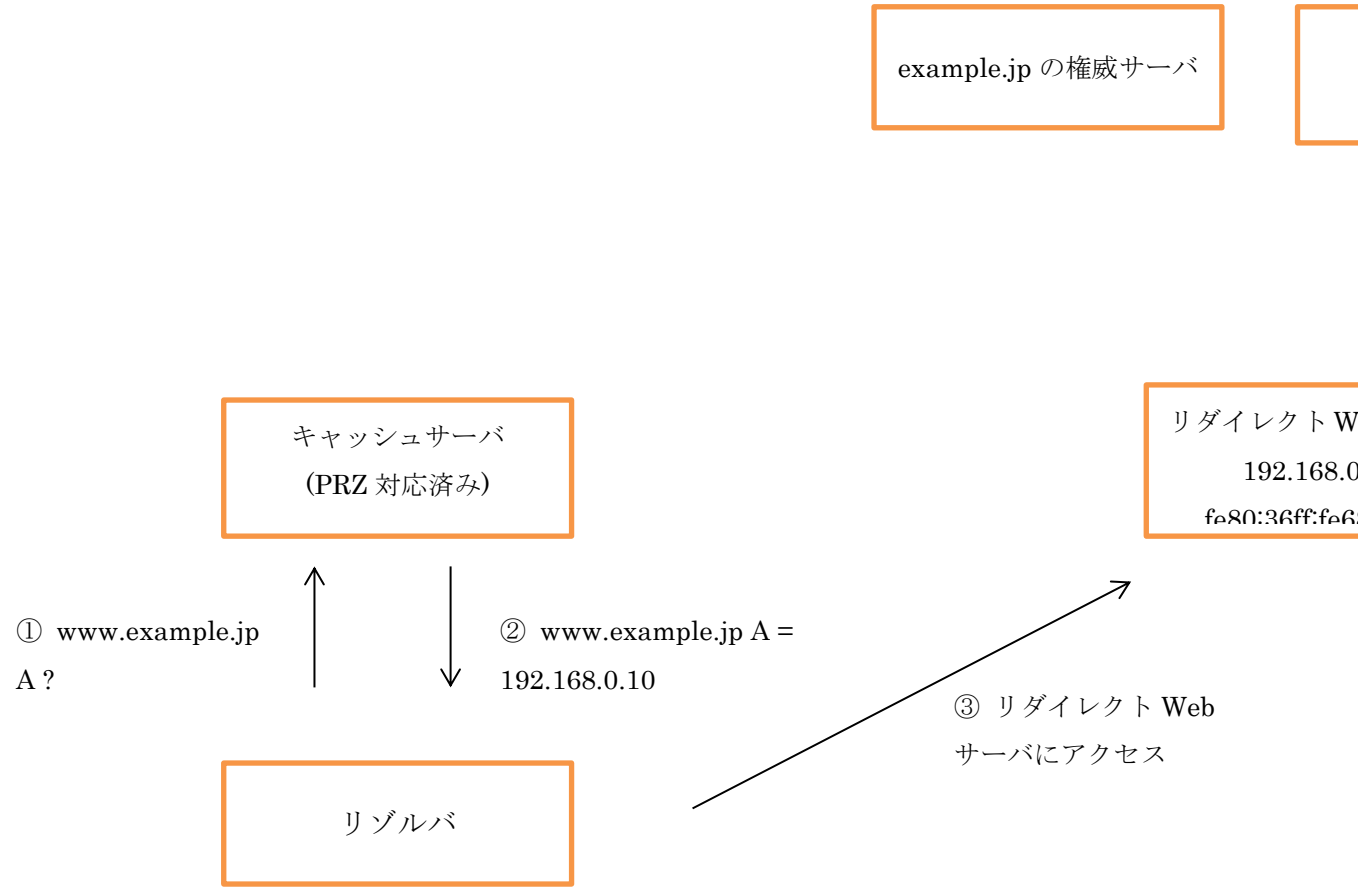


図６　RPZ に対応したキャッシュサーバでブロッキンリストの更新を行う方式

6.2.2　RPZ に対応したリスト配信サーバから RPZ に対応したキャッシュサーバにブロッキングリストを配信する方式

図７にあるように、RPZ に対応したキャッシュサーバに加え、リスト配信サーバ、ブロッキング警告画面として利用するリダイレクト用 Web サーバを準備する必要がある。

リスト配信サーバは、キャッシュサーバにブロッキングリストの配信を行うサーバである。ブロッキングリストの管理はリスト配信サーバ上で実施し、更新したリストは、リスト配信サーバから複数台のキャッシュサーバに対しゾーン転送により、一括配信することが可能である。

リスト配信サーバ 、キャッシュサーバは RPZ を実装している BIND 9.8 以降のバージョンを使用する必要がある。

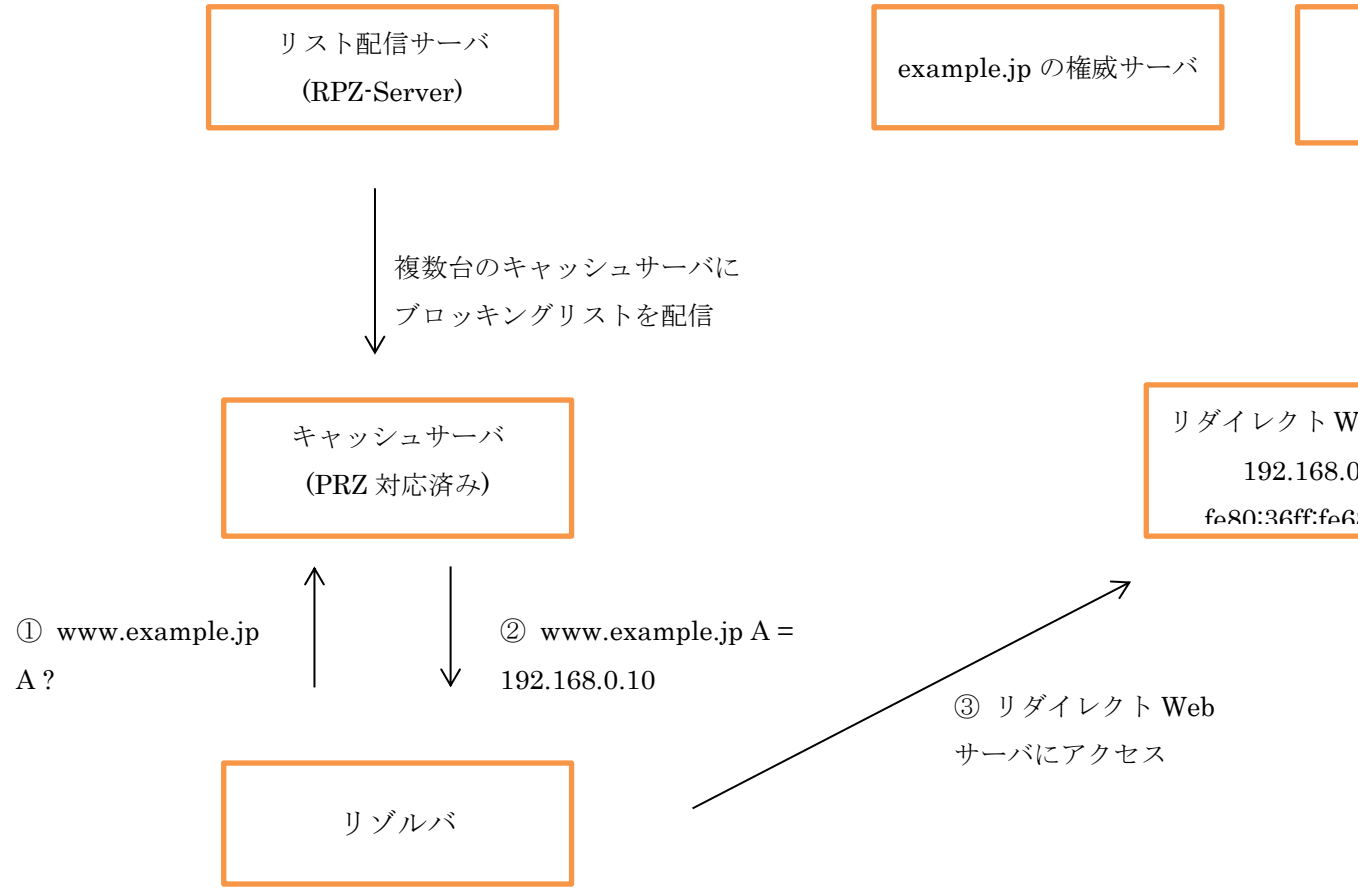


図 7　RPZ に対応したリスト配信サーバから RPZ に対応したキャッシュサーバにブロッキングリストを配信する方式

6.3 設定例

RPZ を使用した具体的な設定方法について説明する。

6.3.1 RPZ に対応したキャッシュサーバ上でブロッキングリストの更新を行う方式

BIND 9.8.1-P1(2011 年 11 月 18 日リリース)を使用し、RPZ の具体的な設定例を説明する。

まず、RPZ に対応したキャッシュサーバに、ブロッキング用のゾーンを登録する。
ここでは、ブロッキング専用のゾーン名を block とする。RPZ で使用するブロッキング用のゾーン名は任意の名前を設定できる。

response-policy ステートメントにブロッキング用の RPZ ゾーン名、zone ステートメントに RPZ で使用するゾーンファイル名を指定する。

ブロッキング用のゾーン block はキャッシュサーバのみが参照し、外部の DNS サーバ、リゾル

バは参照する必要がないため、allow-query、allow-transfer で外部からの参照、ゾーン転送を受け付けないようにアクセス制限を行うのが望ましい。

named.conf

```
options {
        response-policy {
                zone "block";
        };

};

zone "block" {
        type master;
        allow-query { 127.0.0.1; ::1; };
        file "block.db";
        notify no;
        allow-transfer { none; };
};
```

ブロッキング用のゾーンファイル block.db を作成する。block.db ゾーンファイルにブロッキングする FQDN を記述し、DNS ブロッキングを実現する。

まず、ブロッキング用ゾーン block の SOA 、NS 、ネームサーバの A、AAAA レコードを登録する。その次に、ブロッキングするドメイン名、クエリタイプ（ A 、AAAAA ）、リダイレクト先 Web サーバの IP アドレスを記述する。

下記の設定では、リゾルバから、www.example.jp の A または AAAA クエリの名前解決要求があると、RPZ により、リダイレクト先 Web の IP アドレスを返答する。

block.db

```
$TTL 0
@       SOA localhost. root.localhost. (
        01
        1h
        15m
        30d
        2h )
```

```
        IN      NS      ns1.block.

ns1     IN      A       127.0.0.1
        IN      AAAA    ::1

www.example.jp  A       192.168.0.10
                AAAA    fe80::216:36ff:fe68:51e4
```

named プロセスを起動する。

```
# /usr/local/sbin/named
```

シスログより、ゾーン block をロードしていることを確認する。

シスログ

```
named[1768]: zone block/IN: loaded serial 1
```

キャッシュサーバ上で、www.example.jp の名前解決をしたときに、リダイレクト先 Web の IP アドレスが返されるか確認する。

A クエリ

```
# dig @127.0.0.1 www.example.jp a

; <<>> DiG 9.8.1-P1 <<>> @127.0.0.1 www.example.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61343
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
```

```
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

AAAA クエリ

```
# dig @::1 www.example.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @::1 www.example.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28007
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

リゾルバからキャッシュサーバに対し、www.example.jp の名前解決をしたときに、リダイレクト先 Web の IP アドレスが返されるか確認する。

A クエリ

```
リゾルバ# dig @192.168.0.100 www.example.jp a

; <<>> DiG 9.7.3 <<>> @192.168.0.100 www.example.jp a
```

```
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58445
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.
```

AAAA クエリ

```
リゾルバ# dig @192.168.0.100 www.example.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @192.168.0.100 www.example.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10570
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
```

```
ns1.block.              0       IN      AAAA    ::1
```

6.3.2 RPZ に対応したリスト配信サーバから、RPZ に対応したキャッシュサーバにブロッキングリストを配信する方式

RPZ に対応したリスト配信サーバと、RPZ に対応したキャッシュサーバを連携した場合の設定例を説明する。

6.3.2.1 リスト配信サーバの設定

まず、ブロッキング専用のゾーン(任意のゾーン名)を response-policy ステートメント、zone ステートメントに登録する。ここではブロッキング用のゾーン名を block とする。

リスト配信サーバからキャッシュサーバへのブロッキングリストの配信はゾーン転送(AXFR、IXFR)により行われる。ブロッキングリストの配信は、ブロッキングを行うキャッシュサーバのみに限定する。具体的には、also-notify ステートメントで、リストを配信するキャッシュサーバの IP アドレスを記述する。

また、セキュリティの観点から、DNS ブロッキングを行うキャッシュサーバのみから、リスト配信サーバに対し、リスト更新有無の確認要求(SOA 要求)を許可するように設定する。

具体的には、allow-query ステートメントで SOA 要求を許可する IP アドレスを、DNS ブロッキングを行うキャッシュサーバの IP アドレスに限定する。

named.conf (リスト配信サーバ)

```
options {
        response-policy {
                zone "block";
        };

};

zone "block" {
        type master;
        allow-query { 127.0.0.1; ::1; 192.168.0.100; };
        also-notify { 192.168.0.100; };
        file "block.db";
};
```

ブロッキングリストの配信は、セキュリティを考慮し、送信元の認証、通信経路での改ざんを検知できる、TSIG (Transaction Signature)を使用し、リストの配信を行うのが望ましい。

dnssec-keygen コマンドで、リスト配信サーバ上でリスト配信用の TSIG 鍵を作成する。

dnssec-keygen -a アルゴリズム -b 鍵長 -n HOST ゾーン名

```
リスト配信サーバ# dnssec-keygen -a HMAC-SHA256 -b 256 -n HOST block
Kblock.+163+36767
#
```

dnssec-keygen を実行すると拡張子が key 、private の 2 つのファイルが作成される。
TSIG を使用したゾーン転送設定では、拡張子が .key (Kblock.+163+36767.key)を使用する。

```
リスト配信サーバ# ls
Kblock.+163+36767.key  Kblock.+163+36767.private
```

```
リスト配信サーバ# cat Kblock.+163+36767.key
block.  IN KEY 512 3 163 kEIcqwi41+/22Ekh+Ja5NqwwvnxVUZrRN0GrmEjChDI=
```

上記、赤字の文字列(TSIG 鍵)を、鍵名 block-tsig-key として key ステートメントに登録する。
ゾーン block の allow-transfer ステートメントに TSIG 鍵 block-tsig-key を登録し、この TSIG 鍵を保持しているキャッシュサーバに対してのみ、ブロッキングリストの配信を許可するように設定する。

named.conf (リスト配信サーバ)

```
options {
        response-policy {
                zone "block";
        };
};

key "block-tsig-key" {
        algorithm hmac-sha256;
        secret "kEIcqwi41+/22Ekh+Ja5NqwwvnxVUZrRN0GrmEjChDI=";
        };
```

```
zone "block" {
        type master;
        allow-query { 127.0.0.1; ::1; 192.168.0.100; };
        also-notify { 192.168.0.100; };
        ixfr-from-differences yes;
        file "block.db";
        allow-transfer {
                key "block-tsig-key";
        };
};
```

ブロキッキングリスト用のゾーンファイル block.db を作成する。
まず、ゾーン block の SOA 、NS 、ネームサーバの A、AAAA レコードを登録し、その次に、ブロッキングするドメイン名、クエリタイプ（ A 、AAAAA ）、リダイレクト先 Web サーバの IP アドレスを記述する。

block.db（ リスト配信サーバ ）

```
$TTL 0
@       SOA localhost. root.localhost. (
        01
        1h
        15m
        30d
        2h )

        IN      NS      ns1.block.

ns1     IN      A       127.0.0.1
        IN      AAAA    ::1

www.example.jp  A       192.168.0.10
                AAAA    fe80::216:36ff:fe68:51e4
```

リスト配信サーバの named プロセスを起動する。

```
# /usr/local/sbin/named
```

シスログより、ゾーン block をロードしていることを確認する。

シスログ（ リスト配信サーバ ）

```
named[1595]: zone block/IN: loaded serial 1
```

リスト配信サーバ上で、www.example.jp の名前解決をしたときに、リダイレクト先 Web の IP アドレスが返されるか確認する。

A クエリ

```
リスト配信サーバ# dig @127.0.0.1 www.example.jp a

; <<>> DiG 9.8.1-P1 <<>> @127.0.0.1 www.example.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4986
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

AAAA クエリ

```
リスト配信サーバ# dig @::1 www.example.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @::1 www.example.jp aaaa
; (1 server found)
```

```
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58741
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

6.3.2.2 キャッシュサーバの設定

キャッシュサーバの具体的な設定例を説明する。なお、複数台キャッシュサーバが存在する場合は、同様の設定を複数台のキャッシュサーバに設定する。

まず、リスト配信サーバで作成したブロッキング用の RPZ ゾーンと同じ名前(block) のゾーンをキャッシュサーバに登録する。

リスト配信サーバからブロッキングリストの配信を受け取れるように、key ステートメントで TSIG 鍵の登録、masters ステートメントで使用する TSIG 鍵、リスト配信サーバの IP アドレスを指定する。なお、TSIG 鍵、鍵の文字列はリスト配信サーバと同じ鍵を登録する。

キャッシュサーバは、リスト配信サーバのスレーブとして動作するので、type slave とする。

named.conf （ キャッシュサーバ ）

```
options {
        response-policy {
                zone "block";
        };
};
```

```
key "block-tsig-key" {
        algorithm hmac-sha256;
        secret "kEIcqwi41+/22Ekh+Ja5NqwwvnxVUZrRNOGrmEjChDI=";
        };

zone "block" {
        type slave;
        allow-query { none;};
        file "block_slave.db";
        masters { 192.168.0.200 key block-tsig-key; };
        notify no;
        allow-transfer { none; };
};
```

キャッシュサーバの named プロセスを起動する

```
キャッシュサーバ# /usr/local/sbin/named
```

シスログより、TSIG 鍵 block-tsig-key を使用し、ゾーン block のゾーン転送が正常に完了していることを確認する。

シスログ（ リスト配信サーバ ）

```
named[1664]: client 192.168.0.100#34750: transfer of 'block/IN': AXFR started: TSIG block-tsig-key
named[1664]: client 192.168.0.100#34750: transfer of 'block/IN': AXFR ended
```

シスログ（ キャッシュサーバ ）

```
named[1686]: zone block/IN: Transfer started.
named[1686]: transfer of 'block/IN' from 192.168.0.200#53: connected using 192.168.0.100#34750
named[1686]: zone block/IN: transferred serial 01: TSIG 'block-tsig-key'
named[1686]: transfer of 'block/IN' from 192.168.0.200#53: Transfer completed: 1 messages, 6 records, 285 bytes, 0.001 secs (285000 bytes/sec)
```

キャッシュサーバ上で、www.example.jp の名前解決をしたときに、リダイレクト先 Web の IP アドレスが返されるか確認する。

A クエリ

```
キャッシュサーバ# dig @127.0.0.1 www.example.jp a

; <<>> DiG 9.8.1-P1 <<>> @127.0.0.1 www.example.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8406
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

AAAA クエリ

```
キャッシュサーバ# dig @127.0.0.1 www.example.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @::1 www.example.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29406
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA
```

```
;; ANSWER SECTION:
www.example.jp.          0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
block.                   0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.               0       IN      A       127.0.0.1
ns1.block.               0       IN      AAAA    ::1
```

リゾルバからキャッシュサーバに対し、www.example.jp の名前解決をしたときに、リダイレクト先 Web の IP アドレスが返されるか確認する。

A クエリ

```
リゾルバ# dig @192.168.0.100 www.example.jp a

; <<>> DiG 9.7.3 <<>> @192.168.0.100 www.example.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58445
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.jp.                        IN      A

;; ANSWER SECTION:
www.example.jp.          0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                   0       IN      NS      ns1.block.
```

AAAA クエリ

```
リゾルバ# dig @192.168.0.100 www.example.jp aaaa
```

```
; <<>> DiG 9.8.1-P1 <<>> @192.168.0.100 www.example.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10570
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.jp.                        IN      AAAA

;; ANSWER SECTION:
www.example.jp.         0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
block.                  0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.              0       IN      A       127.0.0.1
ns1.block.              0       IN      AAAA    ::1
```

6.4 リダレクト用 Web サーバの設定

リダイレクト用 Web サーバの設定方法は 4.1.3 項を参照。

6.5 動作確認

リダイレクト用 Web サーバを含めたブロッキングの動作確認方法は 4.1.4 項を参照。

6.6 DNS ブロッキング設定により、回答が書き換えられる範囲

RPZ を使用した DNS ブロッキングにより、回答が書き換えられる範囲を以下に記す。

クエリ	クエリタイプ	問い合わせ先	名前解決結果
*.example.jp (www.example.jp は除く)	全てのクエリタイプ	example.jp の権威サーバ	example.jp の権威サーバからの回答
www.example.jp	A	RPZ に対応したキャッシュサーバ	RPZ のゾーンファイルの A
	AAAA		RPZ のゾーンファイルの AAAA
	NS		RPZ のゾーンファイルの NS
	その他		登録されていないレコードの回答は得られない

(注: * は任意の文字列)

6.7 RPZ 方式比較および考察

6.7.1 RPZ を使用しない方式と RPZ を使用する方式について

RPZ を使用しない DNS ブロッキング方式 (方式 1、方式 2)はブロッキング対象ドメインごとにゾーンファイルを作成する必要があるが、RPZ を使用した DNS ブロッキングは、1 つのファイルでブロッキングリストを管理することが可能となる。

6.7.2 RPZ を使用する構成: リスト配信サーバを用意する構成と用意しない構成について

リスト配信サーバを用意せず、キャッシュサーバのみ RPZ に対応させる方式は、ブロッキングリストを更新する際、各キャッシュサーバ上の RPZ ゾーンを更新、named プロセスのリロードが必要である。

一方、配信サーバを用意し、配信サーバからキャッシュサーバへブロッキングリストを配信する方式では、ブロッキングリストの更新は、配信サーバ上のブロッキングリストを更新し、更新

したリストの配信は、配信サーバからキャッシュサーバに対して、ゾーン転送により、自動で配信することが可能である。また、リスト更新時にキャッシュサーバ上の named プロセスのリロードが不要である。

ブロッキングを実施するキャッシュサーバが多数ある場合は、配信サーバを用意し、配信サーバから DNS ブロッキングを実施するキャッシュサーバにブロッキングリストを配信する方式を採用すると、ブロッキングリストの一元管理、一元配信が可能となり、オペレーションが他の方式と比べ容易になることが利点として考えられる。

	リスト配信サーバは用意せず、キャッシュサーバのみ RPZ に対応させる方式	リスト配信サーバを用意し、配信サーバ、キャッシュサーバを RPZ に対応させる方式
ブロッキングリストを保持するサーバ	キャッシュサーバ	リスト配信サーバ キャッシュサーバ
ブロッキングリストの更新作業を実施するサーバ	キャッシュサーバ	リスト配信サーバ
ブロッキングリストの更新時にキャッシュクリアの必要性	必要なし	必要なし
DNSSEC の干渉	ブロッキング対象ドメインへの DNS 問合せの回答は DNSSEC 対応ではない回答がリゾルバに対して返される	ブロッキング対象ドメインへの DNS 問合せの回答は DNSSEC 対応ではない回答がリゾルバに対して返される

6.8 DNSSEC 導入による影響

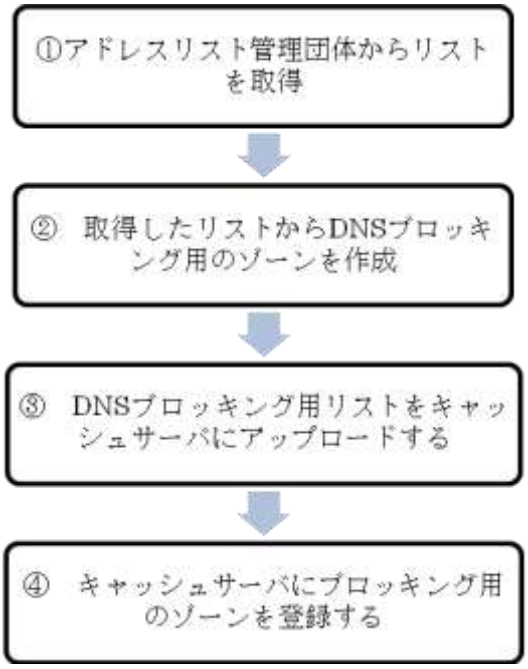
リスト配信サーバを用意せずキャッシュサーバのみ RPZ に対応させる方式も、リスト配信サーバとキャッシュサーバを RPZ に対応させる方式も、DNS ブロッキングによる DNSSEC の影響は、方式 1 と同様の動作となるため、DNSSEC 導入による影響範囲は、5.2.1 項と同様の懸念事項が考えられる。

6.9 導入手順

RPZ を使用し DNS ブロッキングの設定を行うために、いくつかのサンプルスクリプトを用いて、具体的なブロッキングを導入するための設定を説明する。

6.9.1 リスト配信サーバなし　導入全体の流れ

ブロッキングアドレスリストをアドレスリスト管理団体から取得し、そのリストからブロッキング対象ドメインを抽出したリストを DNS キャッシュサーバに登録し、ブロッキング対象ドメインを DNS キャッシュサーバに読み込む。この一連の手順は以下のような流れになる。



6.9.1.1 リスト配信サーバなし 具体的な設定例

① アドレスリスト管理団体からリストを取得

② 取得したリストから DNS ブロッキング用のゾーンを作成

アドレスリスト管理団体より取得したリスト(CSV ファイル)の 2 列目を”掲載ページのホスト名”、5 列目を”掲載ページのブロッキング可否”として、それらの項目を抽出する例について記述する。

awk コマンドで DNS ブロキング可のホスト名のみ抽出したリスト blocking_list_sample.txt を作成する。このコマンドは、リストの 5 列目(掲載ページのブロッキング可否)のフラグが 1(DNS ブロッキング可)のホスト名を抽出する。

```
# awk -F, '$5==1{print $2}' blocking_list_sample.csv > blocking_list_sample.txt
```

awk コマンドを実行すると、下記のリスト(blocking_list_sample.txt)が生成される。

```
# cat blocking_list_sample.txt
```

```
bad.example1.jp
bad.example2.jp
bad.example3.jp
bad.example4.jp
bad.example5.jp
bad.example6.jp
bad.example7.jp
bad.example8.jp
bad.example9.jp
bad.example10.jp
```

③　DNS ブロッキング用リスト(blocking_list_sample.txt)をキャッシュサーバにアップロードする

上記②で作成した blocking_list_sample.txt をキャッシュサーバにアップロードする。セキュリティ上、SFTP、SCP などセキュアな通信でアップロードすることが望ましい。

④　キャッシュサーバサーバにブロッキング用のゾーンを登録する

6.3.1 項で説明したキャッシュサーバの named.conf の設定は完了しているものとする。

次のサンプルスクリプトを用いて、具体的にブロッキング用ゾーンファイルを作成する方法を説明する。

まず、設定用ファイルとして、以下の 2 つのファイルを準備する。

・blocking_list_sample.txt
上記②で作成したブロッキング対象ドメインをリストとして記述したファイル

・make_block_zone.sh
ブロッキング対象用のゾーンファイルを作成するテンプレートファイル。
テンプレートファイルに、RPZ 用ゾーンのネームサーバの IP、TTL 、リダイレクト先 Web の IP アドレスなどについてはあらかじめ記入しておく。

make_block_zone.sh

```
1 #!/bin/sh
```

```
 2
 3 LANG=C
 4
 5 ### usage ###
 6 #
 7 # ./mkzone.sh blocking_list_sample.txt
 8 #
 9 ##############
10
11 DATE=$(date)
12 SERIAL=$(date +"%s")   # Serial UNIX epoch time
13
14 HOSTNAME=localhost.
15 EMAIL=root.localhost.
16 REFRESH=1h
17 RETRY=15m
18 EXPIRE=30d
19 MAXNEGATIVE=2h
20 TTL=0
21
22 RPZ_NS1=ns1.block.
23 RPZ_NS1_IPv4=127.0.0.1
24 RPZ_NS1_IPv6=::1
25
26 WEB_IPv4=192.168.0.10
27 WEB_IPv6=fe80::216:36ff:fe68:51e4
28 FILE=$1
29
30 ###### RPZ SOA ######
31 echo "; File Created  ${DATE}"
32 echo ""
33 echo "¥$TTL ${TTL}"
34 echo "@ IN      SOA     ${HOSTNAME} ${EMAIL} ("
35 echo "                  ${SERIAL}       ; Serial UNIX epoch time"
36 echo "                  ${REFRESH}              ; Refresh "
37 echo "                  ${RETRY}                ; Retry "
```

```
38 echo "                  ${EXPIRE}               ; Expire "
39 echo "                  ${MAXNEGATIVE} )        ; Minimum"
40 echo ""
41
42
43 ##### RPZ Name server info #####
44 echo    "               IN      NS      ${RPZ_NS1}"
45 echo    "${RPZ_NS1}      IN      A       ${RPZ_NS1_IPv4}"
46 echo    "${RPZ_NS1}      IN      AAAA    ${RPZ_NS1_IPv6}"
47 echo    ""
48
49 ##### Blocking List #####
50
51 exec 0<$FILE
52
53 while read -r LINE
54 do
55         echo "$LINE IN A ${WEB_IPv4}"
56         echo "$LINE IN AAAA ${WEB_IPv6}"
57 done
```

スクリプト make_block_zone.sh に下記のパラメータをあらかじめ記入する。

事前に記入するパラメータ	パラメータの内容	サンプル例
HOSTNAME	RPZ の SOA のホスト名	HOSTNME=localhost.
EMAIL	RPZ の管理者のメールアドレス	EMAIL=root.localhost.
REFRESH	RPZ ゾーン情報をリフレッシュするまでの時間	REFLESH=1h
RETRY	RERESH でゾーン情報の更新ができなかった場合に確認するリトライ時間	RETRY=15m
EXPIRE	何らかの理由でゾーン情報のリフレッシュができない状態が続いた場合に、どれだけの期間、ゾーン情報を利用してよいか	EXPIRE=30d

MAXNEGATIVE	存在しないドメイン目の情報のキャッシュ時間	MAXNEGATIVE=2h
TTL	RPZのリソースレコードのTTL	TTL=0
RPZ_NS1	RPZのネームサーバ名	RPZ_NS1=ns1.block.
RPZ_NS1_IPv4	RPZのネームサーバのIPv4アドレス	RPZ_NS1_IPv4=127.0.0.1
RPZ_NS1_IPv6	RPZのネームサーバのIPv6アドレス	RPZ_NS1_IPv6=::1
WEB_IPv4	リダイレクト先Webの IPv4アドレス	WEB_IPv4=192.168.0.10
WEB_IPv6	リダイレクト先Webの IPv6アドレス	WEB_IPv6=fe80::216:36ff:fe68:51e4

make_block_zone.sh に事前の指定しておくパラメータの記入が完了したら、make_block_zone.sh と blocking_list_sample.txt を用いて、ブロッキング用のゾーンファイル tmp_block.txt を作成する。

```
# ./make_block_zone.sh blocking_list_sample.txt > tmp_block.txt
```

以下のようなブロッキング用のゾーンファイルが作成される。

```
# cat tmp_block.txt
; File Created  Mon Feb 27 13:20:48 JST 2012

$TTL 0
@       IN      SOA     localhost. root.localhost. (
                        1330316448      ; Serial UNIX epoch time
                        1h              ; Refresh
                        15m             ; Retry
                        30d             ; Expire
                        2h )    ; Minimum

                IN      NS      ns1.block.
ns1.block.          IN      A       127.0.0.1
ns1.block.          IN      AAAA    ::1
```

```
bad.example1.jp IN A 192.168.0.10
bad.example1.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example2.jp IN A 192.168.0.10
bad.example2.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example3.jp IN A 192.168.0.10
bad.example3.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example4.jp IN A 192.168.0.10
bad.example4.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example5.jp IN A 192.168.0.10
bad.example5.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example6.jp IN A 192.168.0.10
bad.example6.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example7.jp IN A 192.168.0.10
bad.example7.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example8.jp IN A 192.168.0.10
bad.example8.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example9.jp IN A 192.168.0.10
bad.example9.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example10.jp IN A 192.168.0.10
bad.example10.jp IN AAAA fe80::216:36ff:fe68:51e4
```

tmp_block.txt を/var/named/block.db にコピーする。

```
# cp tmp_block.txt /var/named/block_db
```

キャッシュサーバ上の named プロセスをリロードする。

```
# rndc reload
```

キャッシュサーバ上のシスログで、シリアル 1330316448 の RPZ を読み込んだことを確認する。

シスログ(キャッシュサーバ)

```
named[2015]: reloading configuration succeeded
named[2015]: reloading zones succeeded
named[2015]: zone block/IN: loaded serial 1330316448
```

ブロッキング対象のドメインに対し dig により名前解決を実施すると、リダイレクト先 Web サーバの IP アドレス が返されることを確認する。

A クエリ

```
# dig @127.0.0.1 bad.example1.jp a

; <<>> DiG 9.8.1-P1 <<>> @127.0.0.1 bad.example1.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15184
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;bad.example1.jp.                IN      A

;; ANSWER SECTION:
bad.example1.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                      0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.                  0       IN      A       127.0.0.1
ns1.block.                  0       IN      AAAA    ::1
```

AAAA クエリ

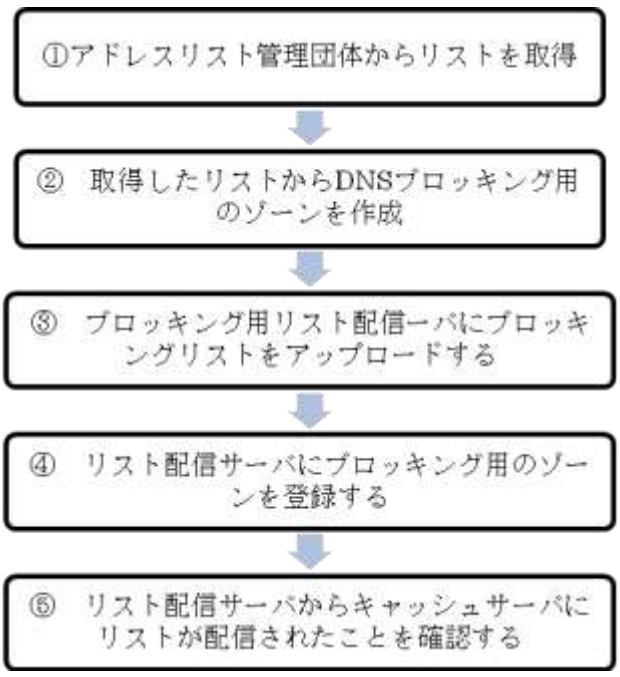
```
# dig @::1 bad.example1.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @::1 bad.example1.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24349
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;bad.example1.jp.                IN      AAAA
```

```
;; ANSWER SECTION:
bad.example1.jp.        0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
rpz.                    0       IN      NS      ns1.rpz.

;; ADDITIONAL SECTION:
ns1.rpz.                0       IN      A       127.0.0.1
ns1.rpz.                0       IN      AAAA    ::1
```

ブロッキングリストの更新は①、②、③、④を実施する。

6.9.2 リスト配信サーバを用意する場合　導入全体の流れ

ブロッキングアドレスリストをアドレスリスト管理団体から取得し、そのリストからブロッキング対象ドメインを抽出したリストについて DNS キャッシュサーバに設定を行い、ブロッキング対象ドメインを DNS キャッシュサーバに読み込む。この一連の手順は以下のような流れになる。



6.9.2.1 リスト配信サーバを用意する場合　具体的な設定例

①　アドレスリスト管理団体からリストを取得

②　取得したリストからDNSブロッキング用のゾーンを作成

アドレスリスト管理団体より取得したリスト(CSVファイル)の2列目を”掲載ページのホスト名”、5列目を”掲載ページのブロッキング可否”として、それらの項目を抽出する例について記述する。

awkコマンドでDNSブロキング可のホスト名のみ抽出したリスト blocking_list_sample.txtを作成する。このコマンドは、リストの5列目(掲載ページのブロッキング可否)のフラグが1(DNSブロッキング可)のホスト名を抽出する。

```
# awk -F, '$5==1{print $2}' blocking_list_sample.csv > blocking_list_sample.txt
```

awkコマンドを実行すると、下記のリスト(blocking_list_sample.txt)が生成される。

```
# cat blocking_list_sample.txt
bad.example1.jp
bad.example2.jp
bad.example3.jp
bad.example4.jp
bad.example5.jp
bad.example6.jp
bad.example7.jp
bad.example8.jp
bad.example9.jp
bad.example10.jp
```

③　ブロッキング用リスト(blocking_list_sample.txt)をリスト配信サーバにアップロードする

上記②で作成したblocking_list_sample.txtをRPZ-Serverにアップロードする。セキュリティ上、SFTP、SCPなどセキュアな通信でアップロードすることが望ましい。

④　リスト配信サーバにブロッキング用のゾーンを登録する

7.3.2 項で説明した リスト配信サーバの named.conf 、キャッシュサーバの named.conf の設定は完了しているものとする。

次のサンプルスクリプトを用いて、具体的にブロッキング用ゾーンファイルを作成する方法を説明する。

まず、設定用ファイルとして、以下の 2 つのファイルを準備する。

・blocking_list_sample.txt
上記②で作成したブロッキング対象ドメインをリストとして記述したファイル

・make_block_zone.sh
ブロッキング対象用のゾーンファイルを作成するテンプレートファイル。テンプレートファイルの中に、RPZ 用ゾーンのネームサーバの IP、TTL 、リダイレクト先 Web の IP アドレスなどについてはあらかじめ記入しておく。

```
# less -N make_block_zone.sh
     1 #!/bin/sh
     2
     3 LANG=C
     4
     5 ### usage ###
     6 #
     7 # ./mkzone.sh blocking_list_sample.txt
     8 #
     9 ##############
    10
    11 DATE=$(date)
    12 SERIAL=$(date +"%s")   # Serial UNIX epoch time
    13
    14 HOSTNAME=localhost.
    15 EMAIL=root.localhost.
    16 REFRESH=1h
    17 RETRY=15m
    18 EXPIRE=30d
    19 MAXNEGATIVE=2h
```

```
20 TTL=0
21
22 RPZ_NS1=ns1.block.
23 RPZ_NS1_IPv4=127.0.0.1
24 RPZ_NS1_IPv6=::1
25
26 WEB_IPv4=192.168.0.10
27 WEB_IPv6=fe80::216:36ff:fe68:51e4
28 FILE=$1
29
30 ###### RPZ SOA ######
31 echo "; File Created  ${DATE}"
32 echo ""
33 echo "¥$TTL ${TTL}"
34 echo "@ IN      SOA     ${HOSTNAME} ${EMAIL} ("
35 echo "                  ${SERIAL}       ; Serial UNIX epoch time"
36 echo "                  ${REFRESH}              ; Refresh "
37 echo "                  ${RETRY}                ; Retry "
38 echo "                  ${EXPIRE}               ; Expire "
39 echo "                  ${MAXNEGATIVE} )        ; Minimum"
40 echo ""
41
42
43 ##### RPZ Name server info #####
44 echo    "               IN      NS      ${RPZ_NS1}"
45 echo    "${RPZ_NS1}      IN      A       ${RPZ_NS1_IPv4}"
46 echo    "${RPZ_NS1}      IN      AAAA    ${RPZ_NS1_IPv6}"
47 echo    ""
48
49 ##### Blocking List #####
50
51 exec 0<$FILE
52
53 while read -r LINE
54 do
55         echo "$LINE IN A ${WEB_IPv4}"
```

```
56          echo "$LINE IN AAAA ${WEB_IPv6}"
57 done
```

スクリプト make_block_zone.sh で下記のパラメータをあらかじめ記入する。

事前に記入するパラメータ	パラメータの内容	サンプル例
HOSTNAME	RPZ の SOA のホスト名	HOSTNME=localhost.
EMAIL	RPZ の管理者のメールアドレス	EMAIL=root.localhost.
REFRESH	RPZ ゾーン情報をリフレッシュするまでの時間	REFLESH=1h
RETRY	RERESH でゾーン情報の更新ができたかった場合に、確認する時間	RETRY=15m
EXPIRE	何らかの理由でゾーン情報のリフレッシュができない状態が続いた場合に、どれだけの時間利用してよいか	EXPIRE=30d
MAXNEGATIVE	存在しないドメイン目の情報のキャッシュ時間	MAXNEGATIVE=2h
TTL	RPZ のリソースレコードの TTL	TTL=0
RPZ_NS1	RPZ のネームサーバ名	RPZ_NS1=ns1.block.
RPZ_NS1_IPv4	RPZ のネームサーバの IPv4 アドレス	RPZ_NS1_IPv4=127.0.0.1
RPZ_NS1_IPv6	RPZ のネームサーバの IPv6 アドレス	RPZ_NS1_IPv6=::1
WEB_IPv4	リダイレクト先 Web の IPv4 アドレス	WEB_IPv4=192.168.0.10
WEB_IPv6	リダイレクト先 Web の IPv6 アドレス	WEB_IPv6=fe80::216:36ff:fe68:51e4

make_block_zone.sh のパラメータの記入が完了したら、make_block_zone.sh と blocking_list_sample.txt を用いて、ブロッキング用のゾーンファイル tmp_block.txt を作成する。

```
# ./make_block_zone.sh blocking_list_sample.txt > tmp_block.txt
```

以下のようなブロッキング用のゾーンファイルが作成される。

```
# cat tmp_block.txt
; File Created  Tue Feb 21 17:30:44 JST 2012

$TTL 0
@       IN      SOA   localhost. root.localhost. (
                        1329813044      ; Serial UNIX epoch time
                        1h              ; Refresh
                        15m             ; Retry
                        30d             ; Expire
                        2h )    ; Minimum

                IN      NS      ns1.block.
ns1.block.        IN      A       127.0.0.1
ns1.block.        IN      AAAA    ::1

bad.example1.jp IN A 192.168.0.10
bad.example1.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example2.jp IN A 192.168.0.10
bad.example2.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example3.jp IN A 192.168.0.10
bad.example3.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example4.jp IN A 192.168.0.10
bad.example4.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example5.jp IN A 192.168.0.10
bad.example5.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example6.jp IN A 192.168.0.10
bad.example6.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example7.jp IN A 192.168.0.10
bad.example7.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example8.jp IN A 192.168.0.10
bad.example8.jp IN AAAA fe80::216:36ff:fe68:51e4
bad.example9.jp IN A 192.168.0.10
bad.example9.jp IN AAAA fe80::216:36ff:fe68:51e4
```

```
bad.example10.jp IN A 192.168.0.10
bad.example10.jp IN AAAA fe80::216:36ff:fe68:51e4
```

tmp_block.txt を/var/named/block.db にコピーする。

```
# cp tmp_block.txt /var/named/block_db
```

リスト配信サーバ上の named プロセスをリロードする。

```
# rndc reload
server reload successful
```

リスト配信サーバ上のシスログで、シリアル 1329813044 の RPZ を読み込んだこと、キャッシュサーバへのゾーン転送が完了したことを確認する。

ゾーンの読み込み完了（ リスト配信サーバ ）

```
named[1713]: reloading configuration succeeded
named[1713]: zone block/IN: loaded serial 1329813044
named[1713]: reloading zones succeeded
```

ゾーン転送の完了（ リスト配信サーバ ）

```
named[1713]: client 192.168.0.100#60209: transfer of 'block/IN': AXFR-style IXFR
started: TSIG block-tsig-key
named[1713]: client 192.168.0.100#60209: transfer of 'block/IN': AXFR-style IXFR ended
```

⑤　リスト配信サーバからキャッシュサーバにリストが配信されたことを確認する

キャッシュサーバ上で、シリアル 1329813044 のゾーン転送が完了したことを確認する。

ゾーン転送の完了（ キャッシュサーバ ）

```
named[1556]: client 192.168.0.200#16941: received notify for zone 'block'
named[1556]: zone block/IN: Transfer started.
named[1556]: transfer of 'block/IN' from 192.168.0.200#53: connected using
192.168.0.100#60209
named[1556]: zone block/IN: transferred serial 1329813044: TSIG 'block-tsig-key'
named[1556]: transfer of 'block/IN' from 192.168.0.200#53: Transfer completed: 1
messages, 25 records, 828 bytes, 0.001 secs (828000 bytes/sec)
```

キャッシュサーバ上で、ブロッキング対象とドメインに対し dig により名前解決を実施すると、リダイレクト先 Web サーバの IP アドレス が返されることを確認する。

A クエリ

```
# dig @127.0.0.1 bad.example1.jp a

; <<>> DiG 9.8.1-P1 <<>> @127.0.0.1 bad.example1.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15184
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;bad.example1.jp.                IN      A

;; ANSWER SECTION:
bad.example1.jp.         0       IN      A       192.168.0.10

;; AUTHORITY SECTION:
block.                      0       IN      NS      ns1.block.

;; ADDITIONAL SECTION:
ns1.block.                  0       IN      A       127.0.0.1
ns1.block.                  0       IN      AAAA    ::1
```

AAAA クエリ

```
# dig @::1 bad.example1.jp aaaa

; <<>> DiG 9.8.1-P1 <<>> @::1 bad.example1.jp aaaa
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24349
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
```

```
;; QUESTION SECTION:
;bad.example1.jp.                IN      AAAA

;; ANSWER SECTION:
bad.example1.jp.        0       IN      AAAA    fe80::216:36ff:fe68:51e4

;; AUTHORITY SECTION:
rpz.                    0       IN      NS      ns1.rpz.

;; ADDITIONAL SECTION:
ns1.rpz.                0       IN      A       127.0.0.1
ns1.rpz.                0       IN      AAAA    ::1
```

ブロッキングリストの更新は①、②、③、④、⑤を実施する。

7. アドレスリスト管理団体とのインタフェース仕様

具体的なアドレスリスト管理団体と ISP との間のインタフェース仕様は、アドレスリスト管理団体であるインターネットコンテンツセーフティ協会にて策定されることになるが、現時点では未確定な部分が多いため、ここでは警察庁事業「官民連携した児童ポルノ流通防止対策に係る調査研究」にて実施したアドレスリスト受渡しの実証実験における仕様から想定した内容について記載する。詳細については、アドレスリスト利用についての申込み方法も含めてインターネットコンテンツセーフティ協会(http://www.netsafety.or.jp)にて確認していただきたい。

7.1 アドレスリストフォーマット仕様

以下の項目を含んだ csv ファイルにて提供される。

① 児童ポルノ画像が掲載されたページのホスト名
② 児童ポルノ画像が掲載されたページの IP アドレス
③ 児童ポルノ画像が掲載されたページの URL
④ 児童ポルノ画像ファイルのホスト名
⑤ 児童ポルノ画像ファイルの IP アドレス
⑥ 児童ポルノ画像ファイルの URL
⑦ 児童ポルノ画像ファイルのハッシュ値
⑧ 児童ポルノ画像が掲載されたベージの IP アドレスの確認年月日
⑨ 児童ポルノ画像ファイルの IP アドレスの確認年月日
⑩ 児童ポルノ画像掲載ページの URL の存在の最終確認年月日
⑪ 児童ポルノ画像ファイルの URL の存在の最終確認年月日
⑫ 事業者への提供年月日
⑬ 児童ポルノ掲載ページのホストに対する DNS ブロッキング対応可否
⑭ 児童ポルノ掲載ページ URL に対する DNS ブロッキング対応可否
等

7.2 リスト受渡方式

アドレスリストファイルが置かれたサーバに対して、セキュリティが確保された方法にて ISP から該当のアドレスリストファイルのダウンロードを行うことによりリストの提供が行われる。

7.3 ブロッキング警告画面

ブロッキング対象のサイトへアクセスが行われた際には利用者に対してはブロッキングが行われた旨を通知するブロッキング警告画面が表示される。ISP 毎にブロッキング警告画面やその掲載内容が異なることは、利用者にとってブロッキングについての内容の理解が得られにくく、広く利用者に周知を行う上でも統一されたブロッキング警告画面がアドレスリスト管理団体により提供される。ブロッキング警告画面には、ブロッキングされた理由、ブロッキングの主旨・目的、問合せ先等が記述される。実際のブロッキング警告画面の設置については、ブロッキング実施主体は ISP であり、それ以外の第３者によりブロッキングが実施されているとの誤解を利用者から受けることを回避するため、ブロッキングを実施する ISP それぞれにおいて設置することとする。

7.4 利用者からの問合せ対応

ブロッキングされたことに対する利用者からの問合せは、アドレスリスト管理団体にて一元的に受付され、対応が行われる。このうち、ISP に係わる問合せについては、アドレスリスト管理団体から ISP へと対応の依頼が行われる。

7.5 サイト管理者からの問合せ対応

自分のサイトが児童ポルノを掲載していないにも関わらずブロッキングアドレスリストに掲載されたと思われる場合には、アドレスリスト管理団体側にて用意された異議申し立てのための受付フォームにて申請を行うことができる。申請された申し立てについては Web サイト上にてアドレスリスト管理団体においての対応状況を確認することができる。

7.6 ブロッキング警告画面へのアクセスログの扱い

ブロッキング警告画面へのアクセスログは、児童ポルノが掲載されたサイトへアクセスしようとした利用者を特定することができるものであり、かつ、アクセスしようとしていた利用者の通信の秘密を形式的に侵害したログでもあるため、その扱いには慎重な配慮が必要である。このような性質を有するアクセスログの保存は違法性が阻却される場合に限り行うことができるものであり、違法性が阻却されない場合にはアクセスログの保存を行ってはならない運用を心がけるべきである。ブロッキングの効果測定を行う等運用上アクセスログの利用が必要となる場合が想定されるが、そのような場合も、違法性が阻却されるか否かを十分に吟味した上で実施すべきであり、その際も利用者の IP アドレスは削除した上で利用する等の慎重な配慮が必要である。

8. 総括

多くの ISP において、導入に際しての障壁の低さからブロッキングの導入方式として期待されている DNS ブロッキング方式について、具体的な設定方法についての解説およびその評価を行ったが、その中でいくつかの課題が明らかになった。１つは性能に関するもので、ISP において提供している DNS のシステム環境によっては、ブロッキング対象のアドレスリストが増大した際にアドレスリストの更新処理において DNS サービスの継続的な提供に影響が発生することがみとめられた。２点目は、DNS のセキュリティ強化のために今後導入が進むことが想定される DNSSEC との関係において、ブロッキングを実装する方法によっては適切にブロッキングの処理ができない場合が発生することである。本来、DNS クエリに対する回答の詐称を防止する技術である DNSSEC と、回答を詐称することを対策としている DNS ブロッキングは相反するものであり、将来的には DNS を利用しない形態でのブロッキング方式に進展していくことも、あるべき方向の一つと考えられる。

また、ブロッキングの実効性を考えた場合においても、DNS ブロッキング方式は限定された範囲でのアドレスリストがブロッキングの対象であり、対象となるアドレスリスト全体に対してのブロッキングとはならないため、さらにきめの細かな画像単位でブロッキングが可能な方式への展開に向けた検討も必要である。今後、さらに精度が高く、かつ、実効性を向上させたブロッキング方式についても、具体的な方式や投資額の算出、効率的な設備構成、運用方法等総合的な観点から検討を進めていくことが必要となる。

ISP とアドレスリスト管理団体とのリスト受渡しやブロッキング実施に関する運用については、今後、実際の運用を進めて行く中で各種の課題が発生することが想定されるが、それらの課題の解決に向けては ISP とアドレスリスト管理団体間にて課題について共有し、解決に向けた協議する体制を構築の上相互に協力しながら、ブロッキングを安定的に、利用者の利便性を低下させることなく運用していくことが重要である。

9. 参考: 商用 DNS 製品を使用した DNS ブロッキングの紹介

参考情報として、商用 DNS 製品 である Infoblox と Nominum を使用した場合の DNS ブロッキングの概要を紹介する。詳細を確認されたい場合は、取り扱いベンダーに問合せ願いたい。

9.1 Infoblox を使用した DNS ブロッキング

Infoblox を使用した DNS ブロッキングの概要について説明する。

9.1.1 Infoblox 会社概要

Infoblox は、1999 年に設立され、40,000 台を超えるアプライアンスを、フォーチュン 500 の 1/3 以上の企業を含む世界の 5,000 社以上のお客様に供給しています。

Infoblox は、DNS,DHCP 等コアネットワークサービス、ネットワーク運用自動化のリーダーです。通信事業者、政府機関、自治体、エンタープライズ等、世界の大手企業の多くが、ビジネスの継続性、可用性、コンプライアンスを実現するために、当社の統合、強化されたアプライアンスベースのソフトウェアを信頼しています。当社では、統合された DNS、DHCP、IP アドレス管理 (IPAM) およびネットワーク変更と構成管理 (NCCM) 製品を提供しています。

また、最高技術責任者にオライリー社の『DNS BIND』の著者である Cricket Liu を中心に製品開発が行われ、且つ、CERT と連携を持ちながら製品作りを行っています。
今回ご紹介する DNS ブロッキングは、日本国内の第一種事業者様にて既に導入頂いております。

9.1.2 Infoblox ソリューション概要

Infoblox は DNS の基本機能であるキャッシュサーバと権威サーバをお客様の用途に応じて一台で構成出来ます。また、より高速なキャッシュサーバを求められるお客様に対応出来る様、高速キャッシュ専用機 (IB-1852, IB-4010) を設けております。どのような用途にも、弊社の特許技術である『GRID 機能』にて、複数台あるサーバを一つの画面にて一元管理が行え、運用コストを最大で 50% 削減できる一方、可用性とアプリケーションのパフォーマンスを強化することが出来ます。
今回ご紹介する『DNS ブロッキング』基本機能となっておりますので、使用する為の特別な装置はライセンスの必要ありません。

GRID機能により、ブラックリストを自動的に複数台のキャッシュサーバに展開する事が可能です。

9.1.3　製品概要

Infobloxの製品は、業種問わず多くのお客様に導入されています。導入のメリットは、DNS, DHCPの機能だけではなく前出のGRID機能にあります。複数台あるDNSサーバを一元管理が行えます。また、複数台のDNSサーバがあっても、１台に対して作業を行う事で、全サーバの運用管理が可能となり、サーバ台数N台分の１まで、運用効率を上げることが可能となります。
また、DNSSECは、導入にあたって設定投入、管理、オペレーションミス等が懸念されますが、Infobloxでは、１クリックで設定することが可能となり、且つオペレーションミスを最低限まで抑える事が実現出来ます。
IPv6は、IPv4/IPv6デュアルスタック、DNS64、DHCPv6は、すべて対応しており、DNSブロッキングもIPv6に対応しています。
Infobloxでは、機能だけではなく、運用の一元化、平準化、効率化を考慮して製品作りを行っております。
また最近では、ネットワークの可視化に焦点を当て、DNS問い合わせの可視化、レイテンシー等の情報をグラフ化する製品もリリースしております。これらにより、ユーザーの接続傾向を可視化し、新サービスへの対応、障害の対応、顧客ニーズへの対応を察知し、ユーザー始動ではなく、サービス事業者始動の対応をとる事が可能となります。

9.1.4　構成例

InfobloxのDNSによるブロッキングを実現する方法としては、キャッシュサーバ上でブロッキングリストを保持する方法をとります。ブロッキングアドレスリストを個々に投入する事も可能ですし、GRIDにて一元管理しキャッシュサーバに配信する配信することが可能です。

9.1.5　キャッシュサーバ上でブロッキングリストを保持する方式の構成例

以下の図の様に、特定ドメインへの問い合わせに対してブロッキングを行い、その通信を警告ポータルサイトに誘導し、そこでブロッキング警告画面を表示させる構成が一般的となります。Infobloxでは、ブロッキングリストの一括投入が可能ですので、投入時の作業を最低限に抑えることが可能となります。

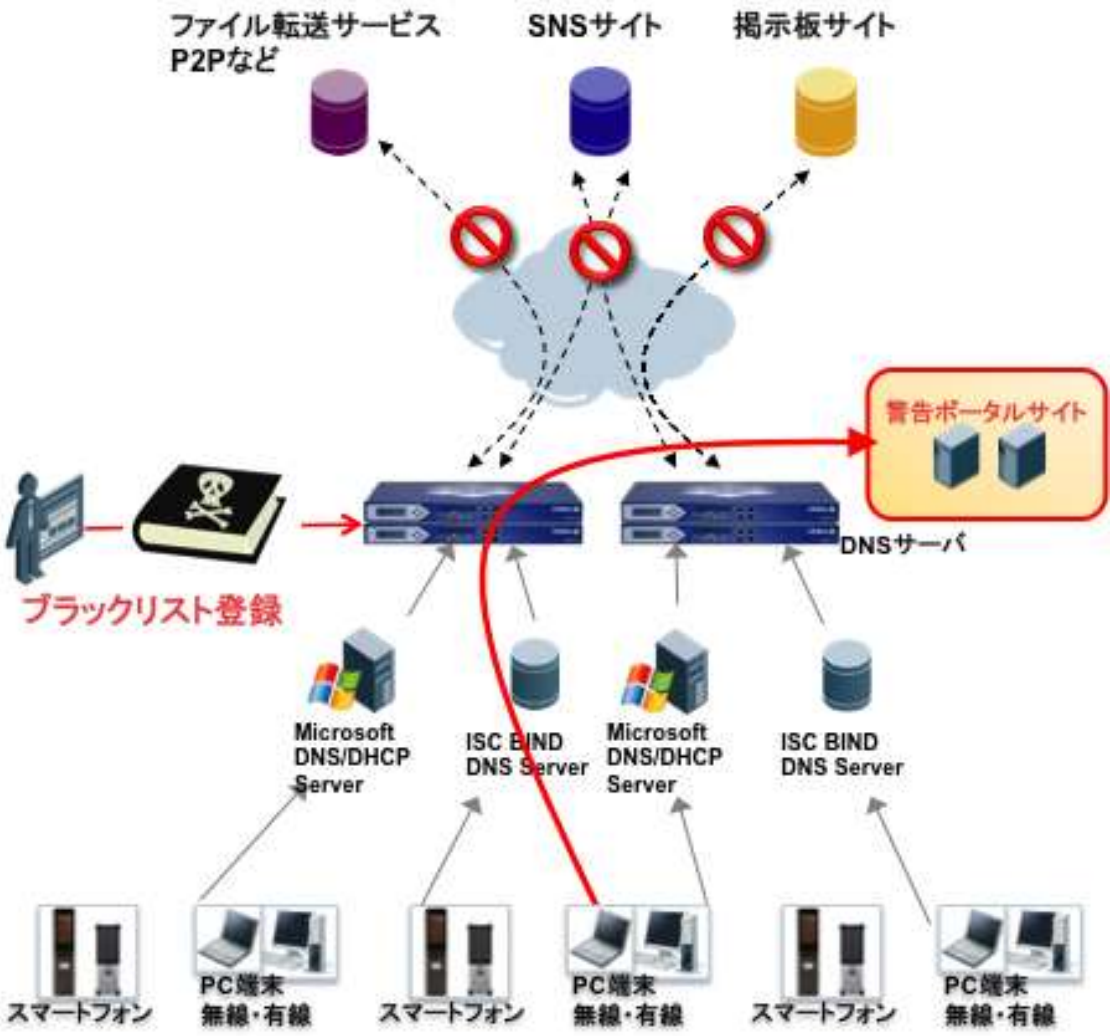


図1　DNSキャッシュサーバでブロッキングリストの構成例

9.1.6　日本国内第一種事業者様での実構成例

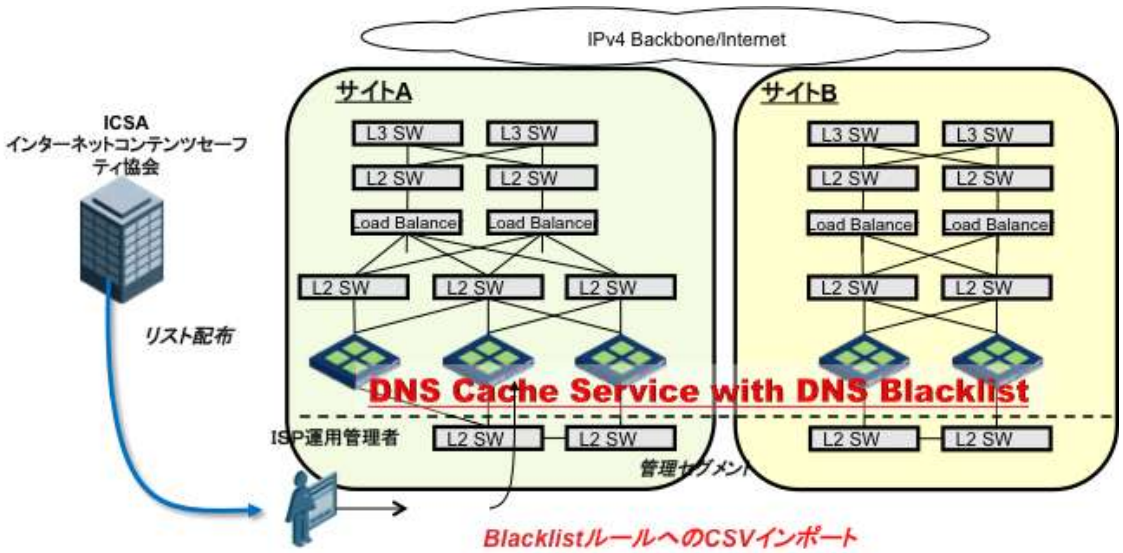


図２　第一種事業者様での実構成例

9.1.7　Infoblox 問い合わせ先

Infoblox 株式会社
営業部：角田
電話：03-5772-7211
Mail: sales-japan@infoblox.com

9.2 Nominum を使用した DNS ブロッキング

Nominum を使用した DNS ブロッキングの概要について説明する。

9.2.1　Nominum　会社概要

Nominum社（本社：米国）はインターネット・ソフトウェア・コンソーシアム（ISC）のBIND9とISC DHCP3を開発するために1999年に設立された。このプロジェクトの経験を活かし、DNSの考案者であり、Nominum社のチェアマンでもあるポールモカペトリス氏を中心に、商用DNS及びDHCP製品を開発、2002年より販売を開始した。

Nominum社は、高いパフォーマンス、高いセキュリティ性を持つ製品を世界各国の通信事業者

をはじめ、企業、大学および政府機関に提供している。DNSの脆弱性に対する強い耐性を持ち、セキュリティを高める機能追加を継続して進めてきた。2002年の製品発売以来、要求の厳しい通信事業者のインターネット環境において安定した稼動を実現している。さらには、DNSブロッキングに対応した製品を2008年より提供しており、一部地域の事業者にて利用されている。

9.2.3 Nominum ソリューション概要

Nominum社はDNSの基本機能であるキャッシュサーバと権威サーバを2つの製品に分けている。これは製品のセキュリティ性を高めるだけでなく、ソフトウェア自体のコード数を減らし、性能の最適化、バグの低減も目的としている。Nominum社が提供するキャッシュサーバはVantio Base Server(以降、Vantio)であり、権威サーバはAuthoritative Name Server (以降、ANS)である。また、DNSブロッキングを実現するキャッシュモジュールとしてMalicious Domain Redirection(以降、MDR)がある。MDRはVantioにインストールするモジュールでの提供となっている。また、ブロッキング対象のアドレスリストを保持・管理するCentrisがある。これらの製品を利用することでDNSブロッキングを実現する。

9.2.4 製品概要

Nominum社が提供するVantioは世界中の多数の事業者に利用されている。キャッシュサーバとして高性能、高可用性といった特徴を持つだけでなく、GUIによるリソースの可視化、SNMPによるDNSサーバの管理、DoS攻撃の検知・遮断機能、A、レイテンシーの最適化を計ることで、快適なインターネットの利用を提供している。また、DNSSEC、IPv6及びDNS64に対応しているだけでなく、APIの装備や、階層化セキュリティにより、キャッシュポイズニング攻撃にも強い耐性を持つ。2002年の製品リリース以降、脆弱性の対応はなく、安定したシステム環境を実現する。

Vantioには複数のモジュールオプションが用意されており、MDRはDNSブロッキングを実現する。MDRはブロッキングリストをキャッシュサーバ上で保持し、ブロッキング対象のドメインあるいはホスト名に対するDNS問合せに対し、リダイレクト先WebサーバのIPアドレスを回答することが可能である。数千万規模のブロッキングリストにも対応でき、本来のキャッシュの性能を劣化させることなく、サービスを提供できる。

さらに大規模な環境に対し、アドレスリストの保持・管理を一元化するリスト配信サーバとしてCentrisがある。これは、複数のアドレスリストを管理・運用していく場合、リストの保管場所を一元化することができ、CentrisからVantio及びMDRに対し、ゾーン転送プロトコル(AXFR、IXFR)でリストの更新を行うことが可能である。また、アドレスリストの外部への流出を防ぐため、CentrisとVantio及びMDR間の通信、Centrisとリスト管理団体との通信を暗号化することが可能である。マニュアルによるアドレスリストの管理・運用にて発生しうる人為的ミスをなくし、自動化したサービスを実現する。

9.2.5　構成例

Nominum社のDNSによるブロッキングを実現する方法としては、BIND及びunboundと同様に①キャッシュサーバ上でブロッキングリストを保持する方法がある。またブロッキングアドレスリストを一括管理しキャッシュサーバに定期的に配信する②アドレスリスト配信サーバにてリストを保持・管理する方法の２つの方法が考えられる。この2つの方法における具体的な構成例を紹介する。

9.2.6　キャッシュサーバ上でブロッキングリストを保持する方式の構成例

example.jpドメイン内のWebサイト www.example.jp (192.168.0.1)へのアクセスに対してブロッキングを行い、その通信をリダイレクト用Webサーバ(192.168.0.10)に誘導し、そこでブロッキング警告画面を表示させる構成が以下となる。図1にあるように構成はBIND及びunboundと同様の構成となる。

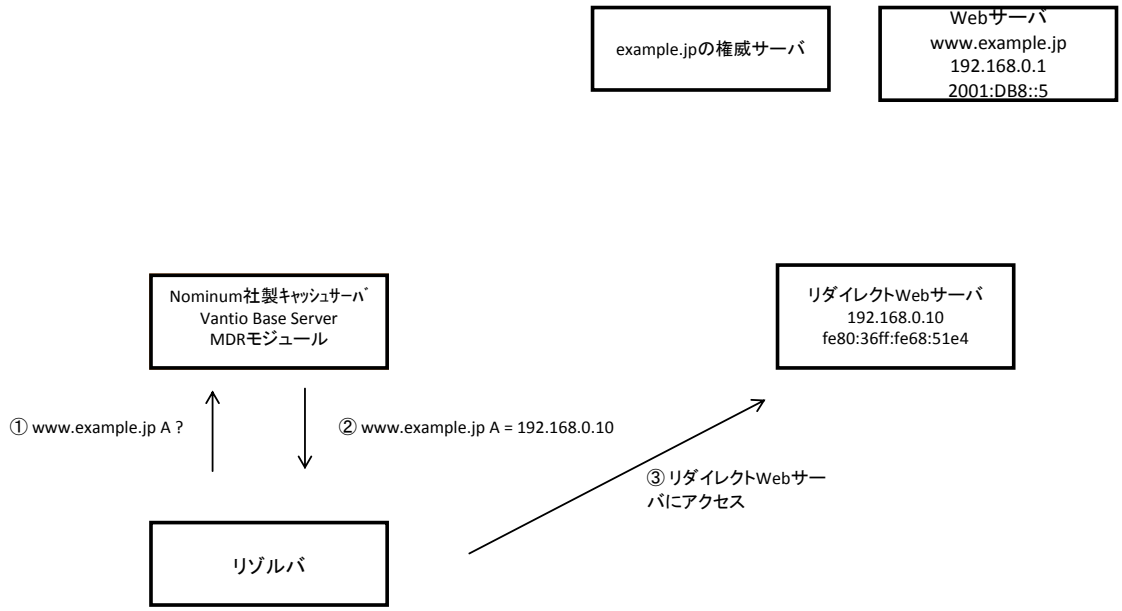


図1　Nominum DNSキャッシュサーバでブロッキングリストを保持する場合の構成例

9.2.7　アドレスリスト配信サーバにてリストを管理・保持する方式の構成例

図2は、ブロッキング対象となるアドレスリストについて一元的に管理し、複数台のキャッシュサーバに対してアドレスリストを定期的に配信する構成である。リスト配信サーバとしてNominum社製Centrisを利用する。複数のブロッキングリストを用いる場合、一旦、Centrisにてリストを管理・保持し、アドレスリストを更新すると、Centrisはキャッシュサーバに対して、ゾーン転送プロトコル(AXFR、IXFR)でリストの配信を行う。CentrisとVantio及びMDR間の通信は全て暗号化される。

キャッシュサーバであるVantio及びMDRはブロッキングのアドレスリスト対象のドメインあるいはホスト名に対するDNS問合せに対して、リダイレクトWebサーバのIPアドレスを回答することで閲覧者に対しブロッキング警告画面を表示させる。

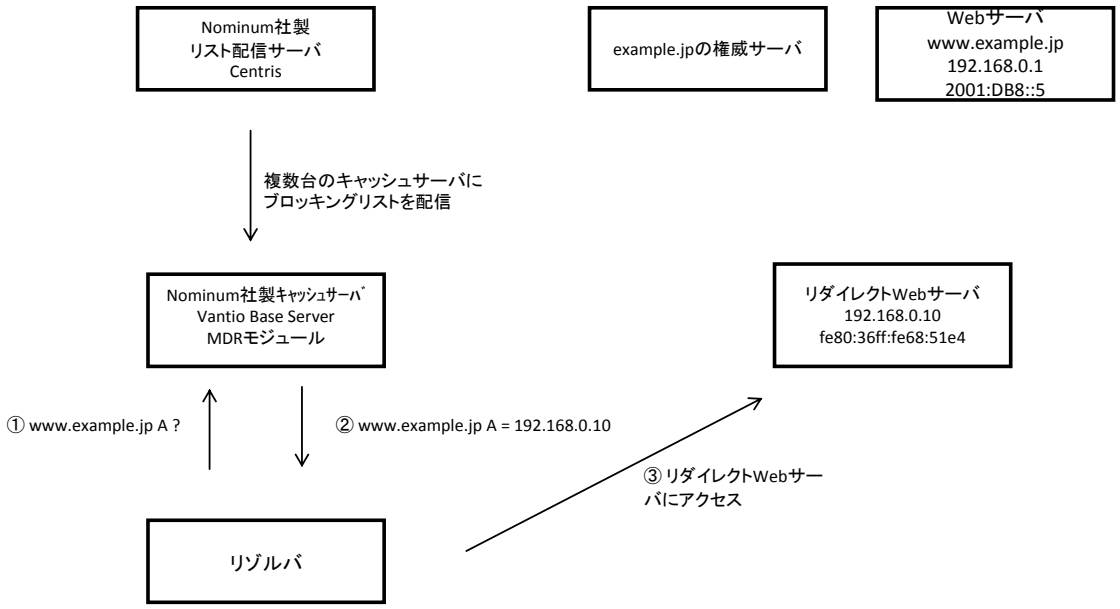


図 2　　アドレスリスト配信サーバにてリストを管理・保持する方式の構成

9.2.8　Nominum 問い合わせ先

E-MAIL：nominum-sales@ml.scsk.jp

SCSK 株式会社

IT プロダクト&サービス事業本部　IP ネットワーク部

Nominum 担当

E-MAIL：nominum-sales@ml.scsk.jp

2011 年度　児童ポルノ対策作業部会 ISP 技術者サブワーキンググループ　構成員

（役職名等は 2012 年 9 月末日時点の記載）

リーダー	北村 和広	NTT コミュニケーションズ株式会社　ネットワークサービス部 担当部長・安心ネットづくり促進協議会　児童ポルノ対策作業部会 副主査
構成員	齋藤　衛	株式会社インターネットイニシアティブ　サービス本部 セキュリティ情報統括室　室長
	山本　功司	株式会社インターネットイニシアティブサービス本部 アプリケーションサービス部　副部長
	岸川 徳幸	NEC ビッグローブ株式会社　執行役員・基盤システム本部長
	持麾　裕之	NECビッグローブ株式会社　経営企画本部 調査シニアエキスパート
	山崎　文生	ソネットエンタテインメント株式会社 システム技術部門プラットフォーム部 IT インフラ課
	泉水 剛志	ソフトバンクBB株式会社 ネットワーク本部技術企画部
	柳舘　一彦	ニフティ株式会社　IT 統括部　基盤システム部　部長
	大はた寿夫	ニフティ株式会社　IT 統括部　基盤システム部　課長
	齋藤　和典	ニフティ株式会社　IT 統括部　基盤システム部　担当課長
	平岡 庸博	KDDI 株式会社 プラットフォーム技術部 インフラプラットフォーム開発グループリーダー
	立石 聡明	社団法人日本インターネットプロバイダー協会 副会長
	明神 浩	社団法人テレコムサービス協会　企画部長
	中島　寛	一般社団法人日本ケーブルテレビ連盟　技術部　部長
	入部 良也	社団法人電気通信事業者協会　調査部 部長
オブザーバー	堀部政男	一橋大学名誉教授・安心ネットづくり促進協議会　会長
	森亮二	弁護士・安心ネットづくり促進協議会　調査研究委員会　委員長 兼　児童ポルノ対策作業部会　主査