DNS サーバの安全性テストのための模擬攻撃プログラムの試作

2012SE180 丹羽 大介 2012SE286 安井 耕平
指導教員：後藤 邦夫

1 はじめに

近年，DNS サーバの脆弱性を突いた攻撃により，個人情報を不正に入手される被害がメディアで多く取り上げられ，社会問題の 1 つとなっている．例えば，2014 年 1 月に中国で広範囲にわたる DNS サービスが停止した事例 [3] や，2008 年 7 月に AT&T の ISP が運営するキャッシュ DNS サーバのキャッシュの一部が書き換えられた事例 [6] がある．これらは，DNS キャッシュポイズニング攻撃 [2] や DNS リフレクター攻撃 [4] により起きたものであると考えられており，常に危険に晒されている．

そこで我々は擬似攻撃プログラムを作成し，閉じたネットワーク内で様々な種類の DNS サーバに攻撃することで，どのような DNS サーバがどのような設定で攻撃を許してしまうのかを検証した．本研究では DNS リフレクター攻撃とカミンスキー攻撃のプログラムを作成し，BIND や NSD，unbound で設定した DNS サーバに擬似攻撃する．プログラミング言語には C++ を使用する．OS が Ubuntu14.04LTS の PC を 1 台使用し，ネットワークエミュレーションソフトウェア「CORE」で仮想ネットワークを構築し，実験する．共同で実験し，丹羽は主に擬似攻撃プログラムの作成を担当し，安井は主にネットワーク構築を担当した．

2 DNS サーバに対する脅威と対策

本節では，DNS サーバに対する脅威を記述する．

2.1 DNS リフレクタ攻撃

DNS リフレクタ攻撃とは，インターネットに接続されているリフレクタ (サーバなど) を攻撃に悪用する手段で，DNS キャッシュサーバと DNS 権威サーバが攻撃に悪用されることがある．攻撃手順を図 1 に示し，リフレクタとして悪用されるものの違い，それぞれの対策を表 1 に示す．

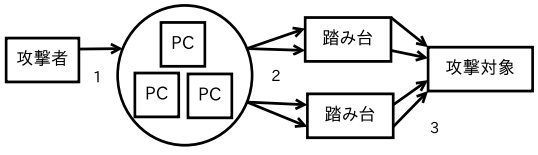

図 1 DNS リフレクタ攻撃

1. 攻撃者が遠隔操作により多数の PC に操作を指令
2. 送信元を攻撃目標に偽装した DNS 問い合わせ (特に大きな TXT レコード) を同時送信
3. 多数の踏み台をリフレクタとして悪用し，攻撃対象へ応答

表 1 DNS リフレクタ攻撃における踏み台と対策

	DNS キャッシュサーバ	DNS 権威サーバ
踏み台	オープンリゾルバ	DNS 権威サーバ
対策	設定修正でオープンリゾルバでなくする．	防御技術の DNS RRL を導入する．

2.2 カミンスキー攻撃

カミンスキー攻撃とは，2008 年 7 月セキュリティ研究者の Dan Kaminsky 氏によって発表された DNS キャッシュポイズニング攻撃の新たな手法である．とあるドメインについて偽装した情報を発信し，DNS サーバに伝えさせることで，利用者がそのドメイン内のサーバに到達できないようにしたり，ドメイン所有者の意図しない別のサーバにアクセスを誘導する．カミンスキー攻撃の手順を図 2 に示す．

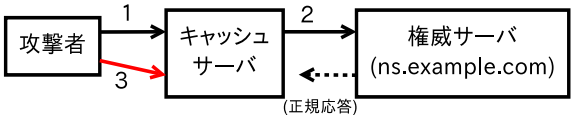

図 2 カミンスキー攻撃

1. <nobody>.example.com の IP アドレスを問い合わせ
2. キャッシュサーバは権威サーバに問い合わせ
3. 正規応答が返る前に txid を偽り，権威サーバになりすまし偽の DNS 応答を送信し，Answer,Additional の情報をキャッシュ

偽の DNS パケットの一例

```
Answer NAME:<nobody>.example.com IP XXX.XXX.XXX
Authority NAME:ns.example.com
Additional NAME:www.example.com IP XXX.XXX.XXX
```

攻撃者は ID の不一致で攻撃できなくても，別のドメイン名で攻撃を試みることで，ID を変化させながら連続して繰り返しキャッシュポイズニング攻撃できる．攻撃への対策は，問い合わせポートを固定せずランダム化することや，DNSSEC の導入などである．

3 擬似攻撃プログラム

本節では，擬似攻撃プログラムの概要について記述する．図 3 はヘッダとセクションにそれぞれどのプログラムにどのような値を入れるのかを示すものである．

	DNS reflector query	Kaminsky attack(query)	Kaminsky attack (response)
IP header			
Src IP	192.168.0.0/16(victim)	any	10.0.0.12(auth)
Dst IP	DNS cache server	DNS cache server	DNS cache server
UDP header			
SrcPort	any	any	53
DstPort	53(domain)	53	Query port(cache)
Header section			
ID		any	brute force
flags	RD(再帰的問い合わせ)	RD	QR(応答) RA(再帰有効) AA(オーソリティ応答)
QDCOUNT	1	1	1
ANCOUNT	0	0	1
NSCOUNT	0	0	1
ARCOUNT	0	0	1
Query section	mail.example.com IN ANY	nobody.example.com IN A	nobody.example.com IN A
Answer section			nobody.example.com IN A 1.1.1.1 (nobody.example.comの誘導先)
Authority section			example.com IN NS ns.example.com
Additional record			mail.example.com IN A 198.162.0.1 (example.comの誘導先)

図 3 Format of Message

3.1 DNS リフレクタ攻撃

DNS リフレクタ攻撃プログラムについて，DNS 問い合わせの UDP パケットのペイロード部分は卒業研究で作成したプログラムを使用した．以下はそのプログラムの一部である．

UDP ペイロード (リフレクタ)

```
ldns_status status
   = ldns_pkt_query_new_frm_str(&query_pkt,
     domain, LDNS_RR_TYPE_ANY,
     LDNS_RR_CLASS_IN, (uint16_t) LDNS_RD);
ldns_pkt_print(stdout,query_pkt);
  fprintf(stdout,"----end pkt_print ---\n");
.....(略).....
status = ldns_pkt2wire(&payload,
         query_pkt, &query_size);
//payload に query_pkt(query_size の大きさ) を配置
```

ldns_pkt_query_new_frm_str() は NSD のライブラリで，指定された名前，種類，クラスの問い合わせパケットを作成する．その後は ldns_pkt_print() でクエリパケットを出力し，ldns_pkt2wire() を status に代入し，正常だと query_size を返す．実行には川上，黒田が作成した DDoSTester.cpp，main.cpp[5] を使用する．DDoSTester.cpp で getRandomV4Address() というクラスを生成することで嘘の Src アドレスを作成する．main.cpp に図 3 のように IP，ポートを設定する．この後は DDoSTester.cpp 内の sendPayload() を呼び出し，ペイロードとペイロードの長さを出力しつづける処理をする．

3.2 カミンスキー攻撃

カミンスキー攻撃プログラムについて，C 言語の公開コード [1] があり，このプログラムを参考に UDP ペイロード部を作成し，DDoSTester.cpp で実行する．このプログラムはクエリ 1 回あたり，ID 総当たり (最大 65536 回) で偽装応答を返し，クエリを変えて，指定回数繰り返す内容である．

query

```
for(r = 0; r = 1; r++){
.....(略).....
 dns_query(payload,sizeof(payload),&len,100,
     DNSF_REC_DESIRED, "nobody.example.com");
 int ret = obj->sendPayload(payload, len);
 cerr << ret << " bytes sent " << endl;
 usleep(100000); // 100 msec
.....(略).....
```

図 3 のように IP とポートを設定し，無限ループ内で DNS キャッシュサーバへ DNS クエリを送る．dns_query() は，Header section のデータをバッファに代入した後に Query section のデータを代入する．また，ペイロードとその大きさ，長さ，ID，フラグ，ドメイン名で構成されており，嘘のドメイン名”nobody.example.com” で問い合わせをする．

response

```
for (s = 0; s < 65534; s++){//ID 総当たり
.....(略).....
 dns_response(payload, sizeof(payload), &len,
  r, DNSF_RESPONSE | DNSF_AUTHORITATIVE,
  "nobody.example.com", "1.1.1.1", "example.com",
  "mail.example.com", "198.162.0.1");
 ret = obj->sendPayload(payload, len);
 usleep(100000); // 100 msec
.....(略).....
```

上で dns_query() をループさせ，試行回数分だけ dns_response() を DNS 権威サーバから DNS キャッシュサーバへ繰り返す．dns_response() は，Header section のデータをバッファに代入した後に，Query section, Answer section，Authority section，Additional record を代入する．また，ペイロードとその大きさ，長さ，ID，フラグ，嘘の問い合わせのドメイン名とその IP，ドメイン，DNS 権威サーバのドメイン名とその IP で構成されており，権威サーバを装って応答する．

4 実験環境の概要

本節では，実験環境の概要について記述する．

4.1 DNS サーバ

本研究では BIND，NSD，unbound を用いる．BIND は，権威とキャッシュの両方の機能があり，BIND1 つで，実験環境が整えることができ，多くの人々が使用している．しかし，カミンスキー攻撃の発覚により BIND だけでは攻撃を防ぐことが難しいことが発覚した．なので，複数の DNS サーバを用いて実験する．NSD はオランダの NLnet Labs が開発している権威ネームサーバで，速度と信頼性と安定性とセキュリティが高く重要である環境における操作のために開発されたものである．本研究では NSD を DNS 権威サーバとしてではなく仮想ネットワーク上での DNS ルートサーバとして使用する．unbound は，キャッシュの機能があり，BIND と比べシンプルな設計で，DNS

キャッシュ汚染に強い運用が可能となっている．

4.2 DNS キャッシュサーバの設定

DNS キャッシュサーバには BIND と unbound を使用することができるが本研究では unbound のみ使用する．unbound の設定としては DNS リフレクタ攻撃に有効と考えられているアクセス制限を加えた場合とオープンリゾルバとした場合で分ける．また，従来の DNS キャッシュサーバは宛先ポートと送信元ポートを 53 に統一していたがキャッシュポイズニングの被害を防ぐために最新版ではデフォルトで送信元ポートをランダムに設定している。本研究では，その送信元アドレスをランダムに設定した場合と攻撃しやすいようにポートを固定した不適切な設定をした場合に分けて実験する．その他には，世界に 13 個ある DNS ルートサーバの情報としてルートヒントがデフォルトで入っているが，実験環境に合わせるためルートヒントの情報を NSD の情報だけにした．

4.3 DNS 権威サーバの設定

前節で述べたとおり，BIND は DNS 権威サーバと DNS キャッシュサーバの両方の機能があるが，本研究では DNS 権威サーバとして BIND を使用する．DNS 権威サーバの情報として BIND のゾーンファイルに以下のような設定を加えた

example.com.zone

```
mail.example.com IN A 172.17.1.100
ns.example.com   IN A 172.17.1.1
```

これらの設定を加えることでそれぞれの IP アドレスに応答するようにした．そのほかにも DNS リフレクタ攻撃を再現するために大きな TXT レコードを用意する．TXT レコードはホスト名に関連付けるテキスト情報を定義するレコードであり，そのテキスト情報をとても長くすることで通信トラフィックを大きくすることで DNS リフレクタ攻撃を再現する．

4.4 システムの概要

本研究は，CORE と DNS サーバを繋いだ仮想ネットワーク環境で実験する．CORE でのネットワーク構成図を以下の図に示す．仮想ネットワークを構築し，セッションを始める．その中で仮想ホストの端末を開き，前節で述べた DNS サーバを稼働させることで，仮想ホスト内にサーバを動かすことができる．また, 図 2 では DNS ルートサーバに NSD，DNS キャッシュサーバに unbound，DNS 権威サーバとして BIND9 を使用している．また，DNS リフレクタ攻撃用の犠牲ホスト (victim) も用意して DNS リフレクタ攻撃対象とする．また，実験環境の IPv4 アドレスを以下のように設定し，ネットワーク構成図を図 4 に示す．

IPv4 アドレス情報

Attacker	:10.0.0.10/24
NSD（DNS ルートサーバ）	:172.16.0.1/24
BIND9（DNS 権威サーバ）	:172.17.1.1/24
unbound(DNS キャッシュサーバ)	:192.168.1.1/24
犠牲ホスト (victim)	:192.168.0.1/24

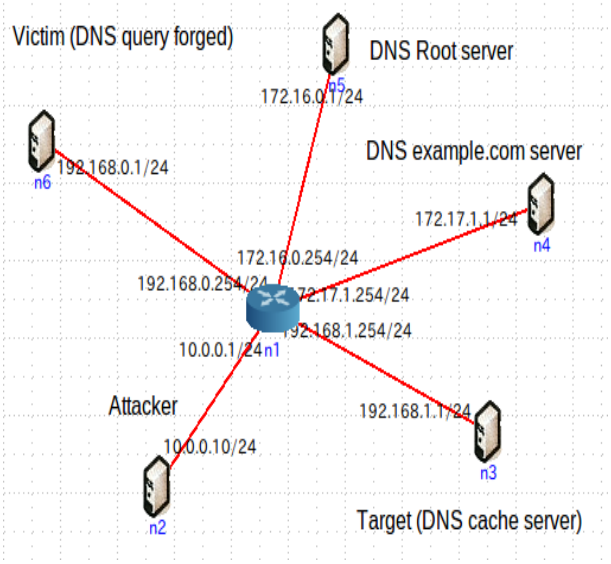

図 4 ネットワーク構成図

5 実験

本節は，実験の手順と実験の結果について記述する．

5.1 手順

仮想ホスト Attacker で攻撃プログラムをペイロードとした UDP 送信プログラムを実行する．

次に攻撃を受けたサーバに対して UDP ポート 53 だけに集中した tcpdump コマンドを使って通信ログの分析する．カミンスキー攻撃に対しては dig コマンドを使用し，DNS サーバの状態を調べる．その結果から毒入れが成功したかどうかを判断する．

その後，通信ログや毒入れ成功の結果によってそのときの DNS キャッシュサーバの脆弱性やセキュリティを評価する．

5.2 結果

DNS リフレクタ攻撃を実行したときの tcpdump の結果を下に示す．

DNS リフレクタ攻撃の tcpdump 結果

```
unbound
IP192.168.17.107.60535>192.168.1.1.domain:1[b2&3
=0x203][1543a][1029q][2057n][2571au][|domain]
IP 192.168.1.1.52593 > 192.168.0.1.domain:
43615+ PTR? 124.62.168.192.in-addr.arpa. (45)

犠牲ホスト (victim)
ARP, Request who-has 192.168.0.1
tell 192.168.0.254, length 28
ARP, Reply 192.168.0.1 is-at
00:00:00:aa:00:08 (oui Ethernet), length 28
IP 192.168.1.1.43136 > 192.168.0.1.domain:
 43615+ PTR? 124.62.168.192.in-addr.arpa. (45)
IP 192.168.0.1 > 192.168.1.1: ICMP 192.168.0.1
udp port domain unreachable, length 126
```

結果を考察すると unbound の 1 行目ではランダムで作られた IP アドレスから unbound へ向けて問い合わせデータなどが送られ，unbound はその応答されたデータを犠牲ホストへ送信している．犠牲ホストの 1 行目から 4 行目ではネットワーク範囲で存在しないアドレスへはルータが ARP 問い合わせに失敗している．5 行目では unbound からデータを受け取っている．7 行目からは UDP port が開いていないので ICMP port unreachable と unbound へ返している．この結果から DNS リフレクタ攻撃が成功していると考えられる．また，アクセス制限を加えると上のような結果にはならず，攻撃を防げていた．よって DNS リフレクタ攻撃にはアクセス制限をつけることで攻撃プログラムの通信を防ぐことには有効であると考えられる．

次にカミンスキー攻撃を実行して tcpdump したものをまとめる．

カミンスキー攻撃の tcpdump 結果

```
IP 10.0.0.10.62846 > 192.168.1.1.domain:
100+ A? nobody.example.com. (46)
IP 172.17.1.1.domain >192.168.1.1.domain:
0*- 1/1/1 A 1.1.1.1 (161)
IP 172.17.1.1.domain >192.168.1.1.45746:
3452 NXDomain 0/1/0 (46)
```

tcpdump 結果から 1 行目には Attacker からキャッシュサーバへ nobody.example.com の問い合わせをしている．その後 2 行目で DNS キャッシュサーバに対して DNS 権威サーバから IP アドレスは 1.1.1.1 だと嘘の情報を応答していることがわかるが，このときは本物の DNS 権威サーバが応答しているのではなく攻撃プログラムが偽の応答をしている．その後 5 行目で本物の DNS 権威サーバからの応答が来ている．その結果の確認として dig コマンドの結果を右上に示す．dig コマンドの結果として 1 行目から 8 行目には問い合わせ時の id などの情報が示されており，9 行目から 10 行目に問い合わせした内容が示されている．11 行目から 12 行目で応答された mail.example.com の IP アドレスが示されている．その IP アドレスが変わってないことから毒入れに失敗したことがわかる．また，DNS キャッシュサーバのポート固定の設定を加えたときでも一緒であった．これは問い合わせ時の id を一致させることができなかったことが原因である．

dig コマンド結果

```
DiG 9.9.5-3ubuntu0.6-Ubuntu <<>> mail.example.com
global options: +cmd
Got answer:
->>HEADER<<- opcode: QUERY, status: NOERROR,
id: 61014 flags: qr aa rd; QUERY: 1, ANSWER: 0,
AUTHORITY: 0, ADDITIONAL: 1
OPT PSEUDOSECTION:
EDNS: version: 0, flags:; udp: 4096
QUESTION SECTION:
mail.example.com.   IN A
ANSWER SECTION:
mail.example.com.   3600   IN   A   172.17.1.100
Query time: 1 msec
SERVER: 127.0.0.1#53(127.0.0.1)
MSG SIZE  rcvd: 45
```

6 おわりに

本研究では DNS 権威サーバに NSD，DNS キャッシュサーバに BIND9，unbound を使用し，DNS リフレクタ攻撃を防ぐにはアクセス制限が適している．また，カミンスキー攻撃について，2 つの DNS サーバで攻撃が防がれたことが確認された．これは、嘘の問い合わせをされたときのキャッシュサーバと攻撃プログラムの応答の間で id，UDP ポートが一致することができなかったからと考えられる．今後の課題としては DNS リフレクタ攻撃の再現をすることと DNS サーバの細かいログを収集することである．

参考文献

[1] Bevand, M.: BIND 9.x - Remote DNS Cache Poisoning Flaw Exploit (2008). https://www.exploit-db.com/exploits/6130/.

[2] IPA セキュリティセンター : DNS キャッシュポイズニング対策　DNS の役割と関連ツールの使い方 (2009). https://www.ipa.go.jp/files/000017289.pdf.

[3] Japan Science and Technology Agency: ドメイン名の解析異常、DNS モニタリングシステムの早期構築が必要 (2014). http://www.spc.jst.go.jp/news/140104/topic_3_03.html/.

[4] JPRS: DDoS にあなたの DNS サーバーが使われる DNS リフレクター攻撃の脅威と対策 (2014). http://jprs.jp/related-info/guide/003.pdf.

[5] 川上健人, 黒田涼介 : DDoS 攻撃模擬ツールの試作, 南山大学システム創成工学科 2015 年度卒業論文 (2016).

[6] 寺田真敏 : 詳細が明かされた DNS キャッシュ・ポイズニングの新手法 (2008). http://itpro.nikkeibp.co.jp/article/COLUMN/20080811/312660/?rt=nocnt/.