An analysis of the DNS cache poisoning attack

Emanuel Petr
CZ.NIC, z.s.p.o.

20 November 2009

Abstract

This document describes the principle of a DNS cache poisoning attack and analyses the factors that positively or negatively influence the attack. It proposes a method for calculating the time necessary for the successful attack with the given probability and applies this method to several attack scenarios carried out in real network conditions (not laboratory) using a freely available implementation of a DNS cache poisoning attack.

Contents

Introduction

The domain name system (DNS) is essential to the overall functioning of the Internet. Potential attackers are also aware of this and therefore try to find weaknesses and ways to attack this system. Since DNS has existed, several attacks have occurred; the latest known functional attack is the variant of DNS cache poisoning discovered by Dan Kaminsky in the summer of 2008. The attack affects recursive caching name servers (NS), whose basic function is to translate a domain name to an IP address. The recommended way of protection was update to a newer version of recursive NS, which already used randomly generated source ports for DNS queries, whereby the threat of attack was significantly reduced. But the question remained as to how long would it take to attack this 'fixed' recursive NS and under what circumstances. The aim of this document is to answer this very question, to indicate the factors influencing an attack, and to offer a methodology for calculating the time requirements of a successful attack with the given probability.

DNS in general

If you are aware of the difference between a recursive and non-recursive NS and know what caching NS means, you can skip this section and move on to the next one - DNS cache poisoning attack theory.

Abbreviations

DNS ..... Domain Name System

NS ..... Name Server (DNS server)

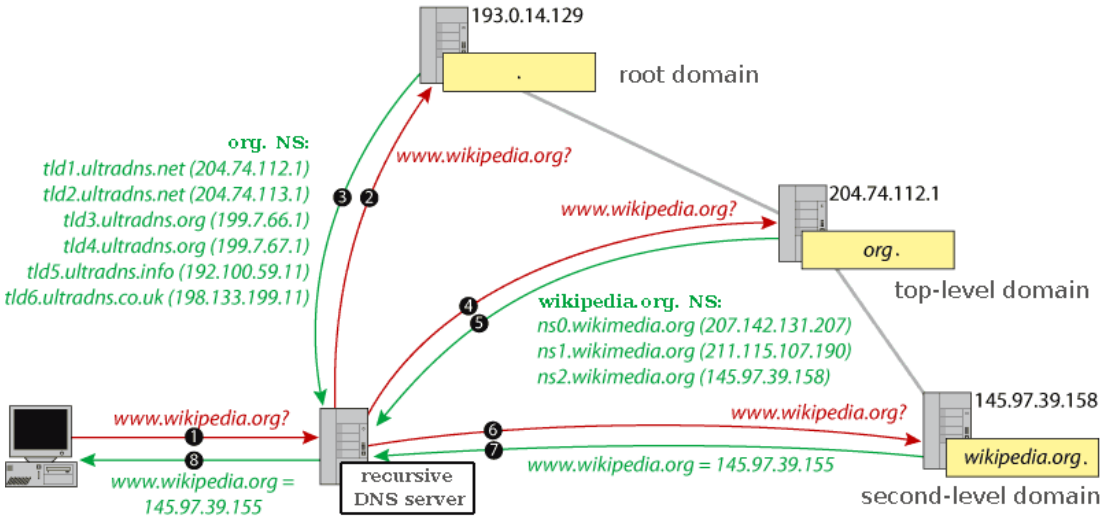
cache ..... temporary data memory

ISP ..... Internet Service Provider

TLD ..... Top-level Domain

Recursive vs. non-recursive NS

DNS was designed in 1983, its basic function is to translate a domain name to an IP address and vice versa. The following image illustrates the translation of a domain name to an IP address.



source: www.wikipedia.org

A **recursive DNS server** performs a full domain translation through the whole domain tree and returns the end result to the client. It is usual that the ISPs provide recursive and caching name servers the their customers.

To improve the efficiency of the translation and reduce DNS traffic, the majority of recursive DNS servers remember the results of their translations by storing them in their internal memory (cache). In case of another already resolved query a recursive search through the whole domain tree is not performed as the information stored in the server memory is used; this behaviour is called 'caching'. The validity or life time of these cached data are configurable for each DNS record and may vary from just seconds to several days or even weeks.

For each recursive NS in this document, we presume this caching feature unless stated otherwise.

A **non-recursive DNS server**, as the name suggests, does not offer the option of a complete translation but provide only 'static' information for which is authoritative. This behaviour is characteristic for the root name servers and TLD name servers. A non-recursive NS has all the necessary information stored in the zone file[i] and therefore does not need to send DNS queries or even store the results of queries, making it immune to a DNS cache poisoning attack. The behaviour of the non-recursive NS is illustrated in the image above for the third and fifth stages of the translation process.

The given description has been simplified, for additional information about DNS see the following web links.

http://www.zytrax.com/books/dns/

http://www.nic.cz/page/312/o-domenach-a-dns/

http://en.wikipedia.org/wiki/Domain_Name_System

(i) Usually as a text file, physically stored on the given NS.

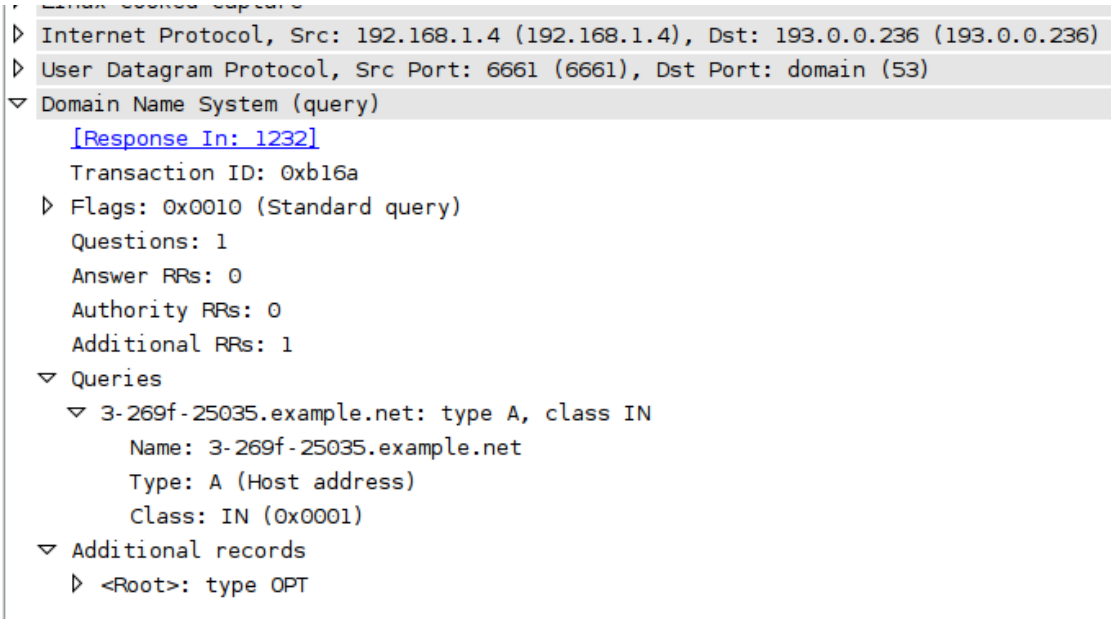
DNS cache poisoning attack theory

A DNS cache poisoning attack tries to forge the response from an authoritative NS thus forcing a recursive NS to store forged information in its internal memory. For this reason the attack is called 'cache poisoning'. As the result of poisoning the recursive NS internal cache all the subsequent queries are resolved[ii] with the forged information.

A forged response must meet these requirements to be accepted by a recursive NS:

- ✔ It is delivered to the recursive NS prior to the response of the authoritative NS.
- ✔ It contains a “Question Section” from the original query of the recursive NS.
- ✔ The identifier (ID[iii]) of the response matches the ID of the query.
- ✔ The source address of the forged response matches the target address of the corresponding query.
- ✔ The target port / address of the forge response matches the source port / address of the recursive NS query.

The following image shows the DNS query to the authoritative NS.

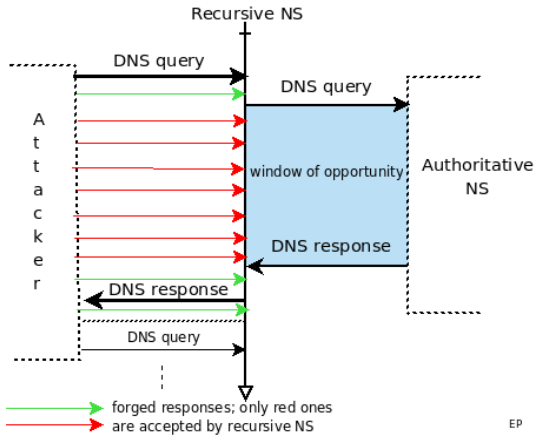


(ii) For the specific life-time of the “poisoned” record.
(iii) The Wireshark application uses for ID field in the DNS header a label ‘Transaction ID’.

The next image shows an attacker's forged DNS response that meets all the requirements listed above.

```
▷ Internet Protocol, Src: 193.0.0.236 (193.0.0.236), Dst: 192.168.1.4 (192.168.1.4)
▷ User Datagram Protocol, Src Port: domain (53), Dst Port: 6661 (6661)
▽ Domain Name System (response)
    Transaction ID: 0xb16a
  ▷ Flags: 0x8580 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  ▽ Queries
    ▽ 3-269f-25035.example.net: type A, class IN
        Name: 3-269f-25035.example.net
        Type: A (Host address)
        Class: IN (0x0001)
  ▽ Answers
    ▷ 3-269f-25035.example.net: type A, class IN, addr 1.2.3.4
  ▽ Authoritative nameservers
    ▷ example.net: type NS, class IN, ns utocnikuv.example.net
  ▽ Additional records
    ▷ utocnikuv.example.net: type A, class IN, addr 1.2.3.4
```

In order for the attacker to start sending forged responses, a recursive NS query must exist for the given domain. Therefore the attacker sends a DNS query to the recursive NS, then a recursive NS finds authoritative NS for the given domain and sends a query to one of them. With this last query a "window of opportunity" is opened for delivering a forged responses and is being closed by the valid response from an authoritative NS. If a forged response is delivered prior to the authoritative NS response and meets all the requirements listed above at the same time, the cache memory of the recursive NS is poisoned and attack is successful.



This type of attack is quite old and, when we leave out of consideration a vulnerabilities in NS that have already been fixed, it is also highly inefficient. Why? Because a recursive NS remembers the result of the response and a further attack is only possible once the validity (life time) of the cached record expires.

In the summer of 2008, Dan Kaminsky discovered a way to make the attack far more effective. Instead of sending queries for resolution of the attacked domain directly, the queries are sent to its subdomains. In our example above where the attack was led on the domain “example.net”, the queries are actually sent to the “XY.example.net”, where XY is a unique string for each query, generated in such a way that it does not collide with an existing subdomain.

How is possible to forged a record for the domain “example.net” when queries are sent to its subdomains?

Two DNS message format sections are used for this purpose - ‘Authority Records’ and ‘Additional Records’. In 'Authority Records' section, a fake authoritative NS for the domain “example.net” is given and in 'Additional Records' section an IP address of this forged authoritative NS is placed. Since the forged response looks like having been sent from an authoritative NS, these added sections will be considered as trusted and accepted by the recursive NS.

The poisoned recursive NS will, as the response to the requests for translation of the domain “example.net” and related subdomains[iv] send queries to the address of the attacker’s fake authoritative NS. The attacker will therefore gain complete control over the translation of the domain “example.net” and all its subdomains.

If the attacker is not able to provide own fake authoritative NS or does not want to do it, the specific name of the forged domain can be given directly in the Authority Records. However, in this way, the attacker loses the ability to control DNS resolution of the other subdomains or to change the address for the given domain name dynamically.

Brute force attack

In the past, a recursive DNS servers were much more vulnerable because of the following weaknesses:

- recursive NS used a static source port for its queries
- recursive NS generated a multiple outstanding queries in case of a multiple

(iv) For example - www.example.net, blog.example.net, etc

identical queries for the same resource record [v]

- ✗ recursive NS used sequential IDs for its DNS queries
- ✗ recursive NS used weak pseudo-random number generator for the query ID

At the end of 2009, the most of a DNS servers introduced measures against the weaknesses listed above and attackers have no other choice but only guess the ID and source port of the query by a brute force, which means to try all the possible values.

Several implementations of the DNS cache poisoning brute force attack appeared on the Internet and they are implemented in such way that try to generate a fake replies as many as possible for the each of the queries send to a recursive NS, this is repeated until corresponding ID and Port is guessed.

In the following sections we are going to analyse the attacker's chances and factors which influence the success of this type of the attack.

(v) http://www.rnp.br/cais/alertas/2002/cais-ALR-19112002a.html

Methodology

For each of the first five test scenarios one thousand requests for domain translation were generated, and from the measured parameters[vi] a duration of the successful attack with the given probability (99%, 95% and 90%) was calculated. For the remaining two scenarios, complete/successful attacks were carried out and the corresponding probabilities were calculated.

Almost[vii] all the tests took place in a real network environment, where each machine was connected via ISP to the Internet exchange point NIX.CZ.

Measurements

The following values were measured for each test:

- length of the 'window of opportunity' (the time between the query of the recursive NS and a genuine response from an authoritative NS)
- the number of a forged responses received within the 'window of opportunity'
- time between answer of the recursive NS and the next attacker's query
- the number of authoritative NS for the attacked domain
- the attacker's bandwidth
- the bandwidth of the recursive NS

Calculations

Time of successful attack (H)

To calculate the time needed for at least one forged response, accepted by a recursive NS, the following formula holds:

$$H = \frac{N}{(1000/W)}$$

H … time requirements of an attack in seconds

N ... number of a 'windows of opportunity' necessary for at least one fake response

W … length of the 'window of opportunity' in ms + overhead for the next 'window of opportunity' in ms

(vi) For details see Measurements section.
(vii) Several exceptions were made in the final test scenario.

The number of 'windows of opportunity' (N) for at least one forged response with the given probability (Q) is calculated according to the formula.

$$N = \frac{\log(1-Q)}{\log(1-P)}$$

Q ... the probability of success (0.95 means 95 %, etc.)

P ... the probability of guessing ID, Port and IP address of the authoritative NS

P is calculated according to the formula:

$$P = \frac{F}{D * U * S}$$

F … number of fake responses received within a 'windows of opportunity'

D … number of possible IDs (0-65535)

U ... number of Ports used (1024 – 65535)

S ... number of authoritative name servers for a domain

The whole formula to calculate the time requirements of successful attack in seconds with the given probability is:

$$H = \frac{\frac{\log(1-Q)}{\log(1-\frac{F}{D*U*S})}}{1000/W}$$

* Probability **Q** is given.
* Values **D, U** and **S** are known.
* variables **F** and **W** are measurable.

The formula only holds when:

- ID and source port of the query (by the recursive NS) is generated randomly
- ID and source port of the forged responses are systematically random numbers

Probability of the attack (Q)

If the duration of the attack is known, the probability (Q) of attack can be calculated using formula[viii]:

$$Q = 1 - (1 - P)^{\frac{T}{W}}$$

P … the probability of guessing ID, Port and IP address of the authoritative NS

T … time requirements of attack in milliseconds (ms)

W … length of the 'window of opportunity' in ms + overhead for the next 'window of opportunity' in ms

Assumptions

Calculations given in this document hold under these assumptions:

- ✔ a domain has two authoritative NS
- ✔ authoritative name servers have only one public address
- ✔ queries are equally distributed between both authoritative NS
- ✔ fake responses are generated for only one authoritative NS address
- ✔ NS implementation BIND9 version 9.4.2 or later is used
- ✔ the average forged DNS response message size is 125 B
- ✔ ID and source port used by the recursive NS are randomly generated numbers within the whole range:

  ID = 0 - 65535

  Port = 1024 - 65535

Note: The quality of the pseudo-random numbers generator (PRNG) is essential; some older versions of the recursive name servers used generator with insufficient randomness. The version of BIND9 used in this document has quite good PRNG; a graphical representation of the ID and source port distribution is provided in the Appendix.

(viii) Based on the formula in section 7.2 RFC 5452, http://tools.ietf.org/html/rfc5452

Comfortable and quick way to find out the randomness of ID and source port offer 'dig' command[ix]:

```
$ dig +short txidtest.dns-oarc.net TXT
$ dig +short porttest.dns-oarc.net TXT
```

If you need to test specific resolver use '@server' argument.

```
$ dig +short porttest.dns-oarc.net TXT @1.2.3.4
```

Used software

DNS cache poisoning attack[x] - slightly modified Evgeniy Polyakov's implementation.

DDoS[xi] attack on an authoritative NS - 'Distributed DNS Flooder v0.1b by Extirpater'.

(ix) https://www.dns-oarc.net/taxonomy/term/7?page=1
(x) http://www.ioremap.net/archive/dns/
(xi) Denial-of-Service

DNS cache poisoning attack testing

Legend

NS ..... Name Server

Hops ..... Number of routers in the path

RTT ..... Round-Trip Time

DNS lookup ..... Time of the domain translation in milliseconds

100 Mbps ..... Bandwidth, 100 Megabit per second

100+ Mbps ..... Bandwidth, 100 Megabit per second and more

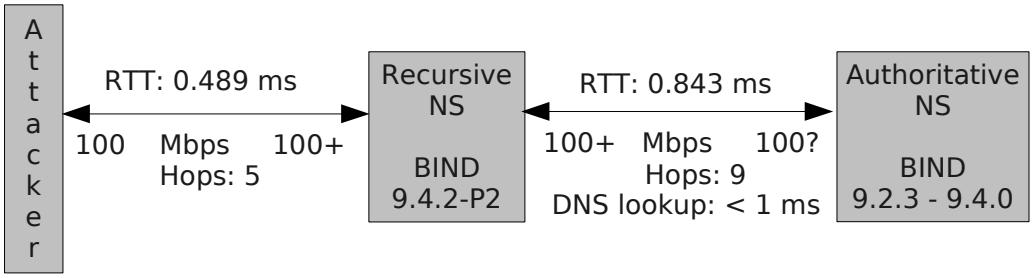
100? Mbps ..... Bandwidth, minimum 100 Megabit per second (max. unknown)

Window of opportunity .... Time between the query of the recursive NS and a genuine response from an authoritative NS

Attack overhead …Time delay between attacks; time when recursive NS does not accepts fake responses

Testing scenario 1 - window of opportunity 1.041 ms

Conditions just before the attack



Measured values

Testing scenario 1	Average	Standard deviation
Window of opportunity	1.041 ms	0.096
Number of fake responses per window of opportunity	57	6
Incoming stream of fake responses	55.05 Mbps	3.86
Overhead per window of opp.	10.451 ms	1.599
Overhead per one second	909.41 ms	

Text description

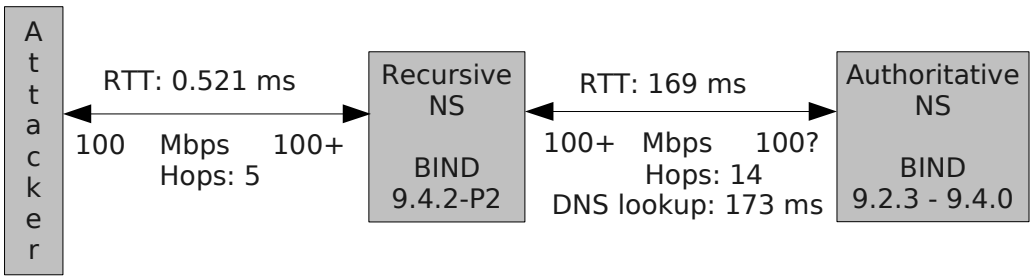
The recursive NS had sufficient bandwidth; round-trip times and DNS lookup times were less than 1 ms. Window of opportunity for getting fake answers accepted was just 1.041 ms per one attack attempt (subdomain resolution) and with rate of 55.05 Mbps of incoming fake responses only 57 of them were delivered within this window. The time delay of 10.451 ms for each window of opportunity together with the fast response times of the authoritative NS resulted in the high attack overhead about 909.41 ms per second of the attack.

Table of success probability

Testing scenario 1 Probability of a successful attack	
99%	2 169 hours (~ 90 days)
95%	1 411 hours (~ 59 days)
90%	1 084 hours (~ 45 days)

Testing scenario 2 - window of opportunity 163 ms

Conditions just before the attack



Measured values

Testing scenario 2	Average	Standard deviation
Window of opportunity	163.678 ms	13.965
Number of fake responses per window of opportunity	8 560	761
Incoming stream of fake responses	52.30 Mbps	2.00
Overhead per window of opp.	3.650 ms	0.592
Overhead per one second	21.81 ms	

Text description

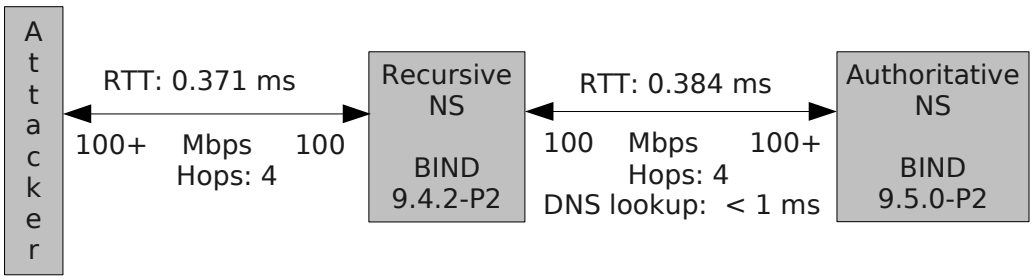
Compared to the first scenario, the authoritative NS had a slower response time, therefore the window of opportunity was 163 ms and attack overhead per second was only 21.81 ms. According to these attributes the time of attack was significantly shorter even though the attacker generated a similar stream of fake responses as in the first scenario.

Table of success probability

Testing scenario 2 Probability of a successful attack	
99%	211 hours (~ 9 days)
95%	138 hours (~ 6 days)
90%	106 hours (~ 4 days)

Testing scenario 3 - the recursive NS link overload

Conditions just before the attack



Measured values

Testing scenario 3	Average	Standard deviation
Window of opportunity	1 291.662 ms	1 518.968
Number of fake responses per window of opportunity	84 236	99 056
Incoming stream of fake responses	65.21 Mbps	0.30
Overhead per window of opp.	11.304 ms	2.676
Overhead per one second	8.676 ms	

Testing scenario 3 Loss rate of DNS messages	
Attacker queries	64.150%
Recursive NS responses	2.631%

Text description

This test scenario is unique in that the link of a recursive NS is overloaded by an attacker generated traffic of 169.19 Mbps. The recursive NS was able to receive only 65.21 Mbps, it meant about 64 % of DNS traffic loss rate, where more of them did not benefit the attacker.

To the attacker disadvantage:

- ✗ Loss of attacker's queries to the recursive NS → a window of opportunity does not open and all fake packets will be ignored
- ✗ Loss of responses from the recursive NS → the attacker will be unable to recognize the window of opportunity and some[xii] traffic will be ineffective

(xii) It depends on to the attacker's query times out (Polyakov's implementation use a five seconds).

- ✗ Loss of attacker's fake responses

To the attacker advantage:

- ✔ Loss of queries from the recursive NS or responses from the authoritative NS → extend a window of opportunity[xiii]

In this scenario, the recursive line overload is counterproductive for the attack and is highly probable that the attack will be detected by the network administrator very soon.

Table of success probability

Testing scenario 3 * Probability of a successful attack	
99%	310 hours (~ 13 days)
95%	201 hours (~ 9 days)
90%	155 hours (~ 6 days)

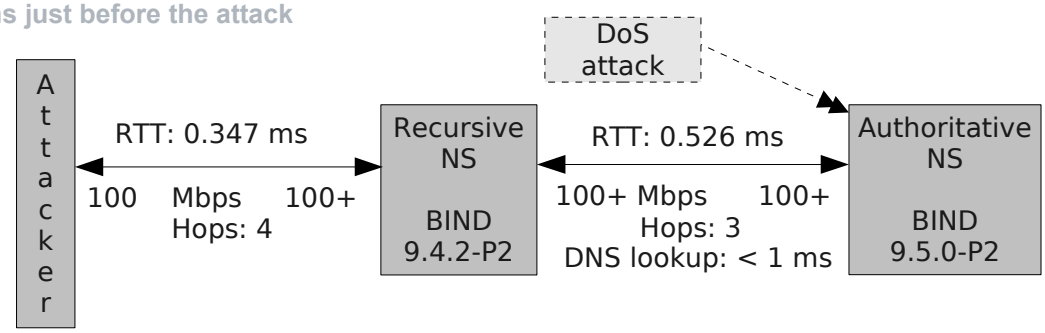
* with 64.15% loss rate of attacker's queries and a five seconds expiration time of the attacker's queries

Note: Due to the load on the bandwidth, the measured average values have large standard deviations, which undermine the calculation of probability.

(xiii) In a range of 0.5 - 10 seconds, see section 'BIND9 and repeated queries'.

Testing scenario 4 - DDoS attack on authoritative DNS servers

Conditions just before the attack



Measured values

Testing scenario 4 without DDoS attack	Average	Standard deviation
Window of opportunity	0.579 ms	0.038
Number of fake responses per window of opportunity	37	4
Incoming stream of fake responses	64.22 Mbps	0.62
Overhead per window of opp.	1.179 ms	0.074
Overhead per one second	670.76 ms	

Testing scenario 4 with DDoS attack	Average	Standard deviation
Window of opportunity	731.930 ms	1 239.457
Number of fake responses per window of opportunity	47 331	80 270
Incoming stream of fake responses	64.67 Mbps	0.36
Overhead per window of opp.	3.519 ms	0.822
Overhead per one second	4.785 ms	

Testing scenario 4 – Loss rate	
Loss rate of authoritative NS responses	4.20%

Text description

In this scenario, the impact of authoritative NS DDoS attack on DNS cache poisoning was investigated.

The DDoS attack was launched on both authoritative name servers. Therefore:

- a queries from the recursive NS had a slower responses
- the window of opportunity was increased
- the overhead of a DNS cache poisoning attack was significantly reduced
- losses of DNS messages between recursive and authoritative NS [xiv]
- the time of the successful attack was approximately three times less than without DDoS on an authoritative DNS servers

Table of success probability

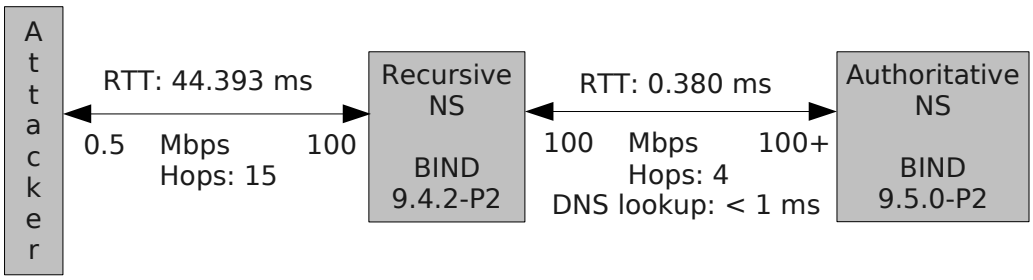
Testing scenario 4 Probability of a successful attack *		
	before DDoS attack	**with DDoS attack**
99%	512 hours (~ 21 days)	145 hours (~ 6 days)
95%	333 hours (~ 14 days)	94 hours (~ 4 days)
90%	256 hours (~ 11 days)	73 hours (~ 3 days)

* with a five seconds expiration time of the attacker's queries

(xiv) That can be considered to the benefit of the attacker.

Testing scenario 5 - home/limited Internet

Conditions just before the attack



Measured values

Testing scenario 5	Wi-Fi	Wired Ethernet
Window of opportunity	0.561 ms	0.655 ms
Number of fake responses per window of opportunity	0 - 3 Average: 0.119	0
Incoming stream of fake responses	0.33 Mbps	0.22 Mbps
Outgoing stream of fake responses	0.41 Mbps	63.10 Mbps
Attacker queries - loss rate	39.6%	98.2%

Text description

The attacker has used a home Internet connection with outgoing traffic limited to 512 Kbps and the attacked recursive NS was outside of ISP network.

Within attacker home network, there were two alternatives for connecting to a home router, the first via Wi-Fi and the second using a wired Ethernet (UTP cable).

Both alternatives delivered a very small number of fake responses within the window of opportunity.

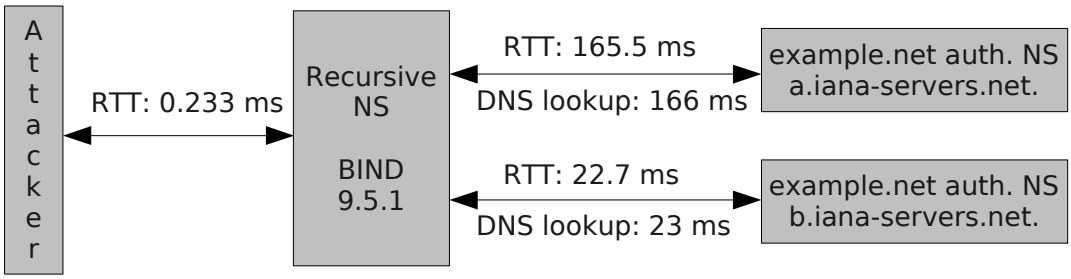
The main reason for the huge loss rate of attacker's queries was limited bandwidth of outgoing traffic. With the wired Ethernet connection the loss of queries reached up to 98.2% and not even one fake response was delivered.

In this scenario, the home/limited Internet connection can't be used for DNS cache poisoning attacks. Another limitation of using home Internet can be existence of spoofed source address detection within ISP network.

Testing scenario 6 - real attack, static source port

Text description

In this scenario, the attack was directed on a recursive NS with static port and tried to forge 'www.example.net' domain record. The analysis of authoritative name servers showed a significant difference in their response times as is described in the following diagram.



BIND 9.5.1 recursive NS contains an algorithm that prefers an authoritative NS with better response[xv], in our case it means 'b.iana-servers.net' with only 23 ms response time compared to 166 ms. Because of this algorithm, all the fake answers in this scenario were sent with the spoofed source address of the authoritative name server 'b.iana-servers.net'.

From a results of a successful attacks, it is evident that a recursive NS with a static source port is highly prone to a DNS cache poisoning attack. Even with a low rate of generated fake responses an attack can be carried out within a few minutes, which falls to a couple of seconds with a rate over 1.5 Mbps.

Attack results

Attack on a recursive NS with a static source port						
Generated traffic of fake responses	Window of opportunity (ms)	Fake responses per window of opportunity	Attack length (seconds)			
			test1	test2	test3	test4
34.16 Mbps	23 - 27	746 - 865	2	1	3	6
10.72 Mbps	19 - 32	202 - 335	3	18	9	8
1.68 Mbps	25 - 26	41 - 42	34	32	7	5
0.56 Mbps	27 - 28	13 - 14	193	76	601	152

(xv) more information in the section 'BIND9 and an authoritative DNS server preference'

Testing scenario 7 - real attack, pseudo-randomly generated source port

Text description

In this scenario, the attack was directed on a recursive NS with pseudo-randomly generated source ports. Usage of a both authoritative name servers of attacked domain was equally distributed with a maximum deviation of 2%, there was no significant loss of DNS messages and other attack conditions met the given assumptions[xvi]. In the last column of the table shown below, the duration of the successful attack together with the calculated corresponding probability is given.

The first two tests were launched in a real network environment, where each server was connected via its provider to IXP NIX.CZ. Since these two tests used a considerable part of the network bandwidth of one of the selected ISPs, the remaining five tests took place within the network infrastructure of CZ.NIC and their purpose was to discover how much the resulting attack probability would differ in case of the same attack conditions.

Attack results

Attack on a recursive NS with a pseudo-randomly generated source port				
Test	Generated traffic of fake responses	Window of opportunity ms	Delivered fake responses per window of opportunity	Duration of the attack (corresponding probability)
1.	85.31 Mbps	45.491	3 820	25 hours 40 minutes (59%)
2.	64.08 Mbps	18.501	1 171	93 hours 41 minutes (88%)
3.	14.34 Mbps	102.241	1 466	64 hours 3 minutes (32%)
4.	14.80 Mbps	684.982	10 139	25 hours 0 minutes (15%)
5.	14.80 Mbps	597.701	8 845	95 hours 52 minutes (45%)
6.	14.15 Mbps	650.851	9 207	50 hours 41 minutes (26%)
7.	14.47 Mbps	504.132	7 293	248 hours 30 minutes (78%)

(xvi) For list of all assumptions see Methodology - assumptions.

BIND9

To carry out an effective attack on a recursive NS it is useful to know the specifics of the given implementation. During our measurements, two interesting (and not so known) behaviour of the BIND9 recursive NS were found.

BIND9 and repeated queries

Each DNS query has specific time for processing (translation) and the query is repeated if this time expires. This behaviour is well documented for a 'Stub Resolver'[xvii] in the manual page 'resolv.conf'. A less known fact is that a 'Full Resolver' also has a algorithm for re-sending queries.

In cases where the recursive NS (Full Resolver) does not get a response, the query is sent again with a repetition interval between 0.5 and 10 seconds. This means that several queries sent by the recursive name server can exist for one query sent by a attacker. Since each re-sent query would have a new value for the ID and the source port, it would increase the chance of guessing them. BIND9 developers have probably realized this potential weakness as they introduced a preventive measure which cancels previous query when the query re-sent occur. This means that a delayed response to the query will not be accepted by a recursive NS.

In real conditions the attack barely uses up all available IDs for one selected source port, therefore a change in ID and Port because of query re-sending will have a negligible impact on the overall calculations of the probability of a successful attack.

The source code of the function '`fctx_setretryinterval`' shows the algorithm used for queries resending by BIND version 9.4.3-P1. [1]

```
static inline void
fctx_setretryinterval(fetchctx_t *fctx, unsigned int rtt) {
   unsigned int seconds;
   unsigned int us;

   /*
    * We retry every .5 seconds the first two times through the address
    * list, and then we do exponential back-off.
    */
   if (fctx->restarts < 3)
      us = 800000;
   else
      us = (800000 << (fctx->restarts - 2));

   /*
    * Double the round-trip time.
```

(xvii) http://tools.ietf.org/html/rfc1123#page-74

```
         */
        rtt *= 2;

        /*
         * Always wait for at least the doubled round-trip time.
         */
        if (us < rtt)
                us = rtt;

        /*
         * But don't ever wait for more than 10 seconds.
         */
        if (us > 10000000)
                us = 10000000;

        seconds = us / 1000000;
        us -= seconds * 1000000;
        isc_interval_set(&fctx->interval, seconds, us * 1000);
}
```

BIND9 and an authoritative DNS server preference

Each domain has at least two authoritative name servers and BIND9 recursive NS has to decide to which one will send a query. BIND9 uses SRTT (Smoothed Round Trip Time) [2] as metric for this decision.

With each query sent by a recursive NS, the SRTT for all authoritative NS of the corresponding domain is updated and a server with a faster response time (lower SRTT) is preferred.

Note that a SRTT for a not actually preferred authoritative NS is automatically lowered with each query, so after a certain number of queries, the previously slower NS is used for translation and its SRTT value is updated according to its current response time.[xviii]

For monitoring SRTT changes this command can be used:

```
$ watch -n 0.5 'sudo rndc dumpdb -cache; sleep 0.5; grep -E
"((192\.0\.34\.43)|(193\.0\.0\.236)).*srtt" /var/cache/bind/named_dump.db'
```

watch -n 0.5	runs command every 500 ms
rndc dumpdb -cache	writes content of the NS cache to the file
sleep 0.5	delay 500 ms[xix]
grep -E "((192\.0\.34\.43)|(193\.0\.0\.236)).*srtt"	prints the matching lines; string

(xviii) The calculation of SRTT can be found in the file lib/dns/adb.c of BIND 9.4.3-P1 source codes.
(xix) In certain cases, the 'grep' command was executed before the contents of the cache could be written to the file; the time delay should prevent this.

containing the “srtt” and also one of the given IP addresses

Output sample:

```
;       192.0.34.43 [srtt 39866] [flags 00000000] [ttl 1711]
;       193.0.0.236 [srtt 21929] [flags 00000000] [ttl 1711]
```

The attacker can take advantage of this behaviour and generate fake DNS replies with a source address of the authoritative NS with a better response times. It will result in a better probability of guessing the address of the authoritative NS because the recursive NS will also prefer a “faster” authoritative NS.

As a demonstration of an authoritative NS preference the resolution of the domain “example.net” was chosen. The difference in response times, as you can see in example below, is about 133 milliseconds.

```
$ dig NS example.net
...
;; ANSWER SECTION:
example.net.            172800  IN      NS      a.iana-servers.net.
example.net.            172800  IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
b.iana-servers.net.     16660   IN      A       193.0.0.236
a.iana-servers.net.     163313  IN      A       192.0.34.43


$ dig test.example.net @192.0.34.43 | grep "Query time"
;; Query time: 158 msec


$ dig test.example.net @193.0.0.236 | grep "Query time"
;; Query time: 25 msec
```

Four tests were launched, for each of them, a thousand requests for translation of the domain ‘example.net’ were sent. The results show the clear preference of an authoritative NS with better response times.

Authoritative NS preferences				
193.0.0.236	**98.1%**	**98.1%**	**98.1%**	**98.1%**
192.0.34.43	1.9%	1.9%	1.9%	1.9%

Discussion

Attack conditions

The DNS attack efficiency is influenced by many factors. If an attacker is not aware of the restrictive circumstances, the attack can be run completely without effect. On the other hand, network or name server administrators can use these factors to restrict or prevent an attack.

Limiting factors of a DNS cache poisoning attack		
Name	Impact	Description
Spoofed source address detection	An attack is not possible	The router between the attacker and the recursive NS blocks packets with a spoofed source address
Recursive and authoritative NS with DNSSEC	An attack is not possible	DNS Security Extensions[xx] protect against DNS cache poisoning attacks
Recursive NS with query filtering	Restricts an attack from permitted networks	The NS accepts recursive queries only from the list of permitted networks[xxi]
Random ID and source ports	Reduces the effectiveness of an attack	A recursive NS uses randomly generated ID and source ports for its queries
More authoritative NS for the domain	Reduces the effectiveness of an attack	Each new authoritative NS[xxii] reduces the probability of guessing the source address
IPv6 address on an authoritative and recursive NS	Reduces the effectiveness of an attack	IPv6 address on an authoritative and recursive NS reduces the probability of guessing the response source address of the authoritative NS
Fast response time / high-capacity bandwidth of an authoritative NS	Reduces the effectiveness of an attack	Fast response times reduce a window of opportunity. Sufficient bandwidth makes DDoS attacks on an authoritative NS more difficult.
Insufficient router performance	Reduces the effectiveness of an attack	Insufficient router performance causes loss of attacker's traffic[xxiii]

(xx) For more information visit http://www.nic.cz/dnssec/ and http://www.dnssec.net/
(xxi) For BIND9 using 'acl' and 'view' see http://www.zytrax.com/books/dns/ch7/acl.html
(xxii) With similar response time, see 'BIND9 and an authoritative DNS server preference'
(xxiii) See 'Testing scenario 5 - home/limited Internet'

Insufficient bandwidth of the attacker or recursive NS connection	Reduces the effectiveness of an attack	Line congestions between the attacker and the recursive NS causes traffic loss [xvii]
Server or network monitoring	Blocking an attack	Monitoring can recognise increased data flow or load on system resources

Factors that support an attack whose opposite is not given above:

Supporting factors of a DNS cache poisoning attack		
Name	Impact	Description
DDoS against the authoritative NS	Increases the effectiveness of an attack	The DDoS attack delays or suppresses the response of an authoritative NS and thereby extends the window of opportunity for fake traffic and reduces attack overhead
Recursive NS behind address translation	Increases the effectiveness of an attack	NAT can degrade the source port randomness of a recursive NS
Significant difference in the response time of authoritative NS	Increases the effectiveness of an attack	A recursive NS prefers an authoritative NS with better response time; if the attacker does the same, there will be a better chance of guessing the source address of the fake DNS response[xxiv]

Calculation methodology

The absence of detailed analysis of the probability of DNS cache poisoning attack on recursive NS with random source ports was the main motivation for this document. Some older analyses can be found on Internet but their results can't be used because of obsolete assumptions, such as the usage of a static ports or 'Birthday Attack' method.

A good source of information provides document RFC 5452 [3], in section 7.2, a calculation is given which is quite similar to the calculation we have used, but still works with the TTL. Now, we know that TTL can be excluded from the formula and, on the contrary, we must take into account the randomness of the source port. Another variable which should be taken into account is the overhead per window of opportunity. This overhead can significantly affect the resulting probability as can be seen by comparing results of the first two testing scenarios.

Even if a more advanced statistical method has not been applied to the calculation,

(xxiv) See the chapter 'BIND9 and selection of an authoritative DNS server'

a results of the performed attacks in this document give us a general idea of the attackers' chances and based on the final seven attacks, we can say that the calculated time requirement of the attack is generally higher than will be actually necessary.

Conclusion

An attack on a recursive NS with a randomly generated source ports is far more complicated than in the case of a static ports, but it is still possible. The attack can be successful within several days or weeks, however, it is necessary to generate a data stream of fake responses in order of Mbps; the specific value depends on the response times of the recursive NS and other factors as described in the section 'Attack conditions'. From a technical point of view, it is not a problem to generate the necessary traffic, but watchful network or server administrator can identify the attack by detecting unexpected incoming stream of DNS messages.

But since monitoring and the human factor can fail, a reliable solution against DNS cache poisoning attacks is the introduction of DNSSEC technology, which provides data integrity and the certainty that they have been provided by the correct source.

References

[1] ISC BIND version 9.4.3-P1 source code
ftp://ftp.isc.org/isc/bind9/9.4.3-P1/bind-9.4.3-P1.tar.gz

[2] Karn, Phil., Partridge, Craig., *Improving Round-Trip Time Estimates in Reliable Transport Protocols*, 1987, ACM SIGCOMM
http://www.ka9q.net/papers/rtt.ps.gz

[3] Hubert, A., R. van Mook, *Measures for Making DNS More Resilient against Forged Answers*, RFC 5452, January 2009
http://tools.ietf.org/html/rfc5452

Appendix

BIND9 – a distribution of the source ports and message IDs

The graph displays values for one thousand queries sent by the recursive NS.