DNS Cache Snooping

or

Snooping the Cache for Fun and Profit

Version 1.1 / February 2004

Luis Grangeia

luis.grangeia@sidestep.pt

Abstract

This research paper presents a technical overview of the technique known as DNS cache snooping. Firstly, a brief introduction to DNS is made followed by a discussion on common misconceptions regarding DNS sub-systems. Then this relatively unknown technique is introduced, followed by a field study to assert the overall exposure of the Internet to this threat. Also, a set of devised abuse scenarios that rely on cache snooping is presented. This paper concludes with recommendations on how to reduce exposure to this problem, including proposed changes to the BIND DNS server implementation.

Introduction

One of the most critical aspects of information security is awareness. One must be conscious of all possible security weaknesses, however inconsequent they may seem when studied in isolation. In any network where it is to be taken seriously, security is achieved by combining different countermeasures that work in parallel to try to eliminate any "single point of failure" that compromises security. On the other end of the network, a serious attacker, lacking an obvious "single point of failure", will try to combine all the seemingly inconsequent weaknesses to bypass security and successfully achieve his goals.

Being DNS the "invisible engine" behind just about every service available on the Internet, it seems pertinent to raise awareness on a somewhat unknown information disclosure vulnerability known as DNS cache snooping and its implications.

The author found that discussion on this subject is scarce, amounting to a few emails found on mailing lists and newsgroups. No document presenting a concise definition was found, let alone explaining the several ways of doing it or its security implications. It seems that there is general unawareness and diverse misconceptions around this subject.

DNS Overview and Some Controversy

The DNS, or Domain Name System, is an elegantly designed distributed and fault-tolerant database. This database contains, among other things, information on domains and hostnames and how they relate to the IP addresses that are assigned to the various computer systems that compose the Internet.

As with any directory-based service, there are two main user approaches to the DNS system: the Publisher, who wants to make his information available to others to look up; and the Browser who queries the system for information for his personal use. This duality is transposed to the DNS architecture as a natural separation of functionality, where sometimes a DNS system will function as a cache and store recently queried information to optimize further local queries, and on other occasions the system will serve

information about hostnames and/or IP's that it is responsible, or "authoritative", about. As each of these "use cases" has different objectives, it makes sense to compare both in regards to their requirements.

Following is a table describing the fundamental differences between DNS caches and servers.

	DNS Server	**DNS Cache**
Availability	Should be able to respond to queries from all around the Internet	Should only respond to queries that originate from a local user base
Types of query that it should answer	non-recursive queries	recursive queries
Design of a software implementation	Small, stateless, optimized for speed, no internal cache	Must maintain state because of recursive queries and an internal cache
Records that it should attempt to resolve	Should only respond data that it is authoritative about	Should attempt to resolve any request

The need to separate "DNS servers" and "DNS caches" has been the source of controversy and discussion [1] [12]. Some say that caches and servers should be run as independent services on different IP addresses, while others maintain the opposite can also be secure.

The fact that most programs that implement DNS functionality bundle both functions into one software package helps to this confusion. The most obvious example of this is the BIND (Berkley Internet Name Domain) software. This DNS package allows for a very diverse set of configurations, and it is indeed possible to configure it in a way that separates the DNS Server and Cache functions to some degree, but issues remain to be resolved. These issues, as well as an example for a relatively secure BIND configuration will be looked into in the final sections of this paper.

The fact that BIND can be configured securely does not mean that this is the norm. In fact, a small study presented later in this paper shows that the opposite is true for the majority of the servers tested.

Therefore, while it may not be necessary to create different programs for the purpose, the author maintains that a logical separation is needed, as it can greatly improve the security of the DNS infrastructure.

DNS Cache Snooping

DNS cache snooping is not a term the author just made up, it is known and discussed by some notable DNS implementation developers, and a few interested DNS administrators have probably at least heard of it.

After stumbling into the technique by doing research with the "dig" program [2], the author set out to the Internet, in an attempt to uncover documentation about the subject, explaining what it is and ways to exploit it.

Only two references to the subject were found available. Dan Bernstein, the creator of the djbdns software package refers to the problem in his Website [3]:

> "`dnscache` tries to prevent local users from snooping on other local users. It discards non-recursive queries; it discards inverse queries; and it discards zone-transfer requests. If $HIDETTL is set, `dnscache` always uses a TTL of 0 in its responses. In versions before 1.03, `dnscache` always uses a TTL of 0 in its responses."

Also, an email from Rob Mayoff [4] to the djbdns mailing list refers the problem by mentioning the technique in more detail:

> "If dnscache answered non-RD queries strictly from cached answers, then I could try to find out whether other users have asked about domain X by sending a non-RD query for domain X. I can still snoop less reliably by measuring dnscache's response time. Obviously if I am the admin, or have access to the dnscache logs, then I have much greater snooping power."

DNS cache snooping is then the process of determining whether a given Resource Record (RR) is (or not) present on a given DNS cache. There are two ways of accomplishing this, and they are presented as follows.

The Technique – The Ecological Way (Non-recursive Queries)

The most effective way to snoop a DNS cache is using iterative queries. One asks the cache for a given resource record of any type (A, MX, CNAME, PTR, etc.) setting the RD (recursion desired) bit in the query to zero. If the response is cached the response will be valid, else the cache will reply with information of another server that can better answer our query, or most commonly, send back the root.hints file contents.

Here is an example of using iterative queries to snoop a DNS cache. Output is abbreviated for illustrative purposes:

```
$ dig @ns1.tvcabo.pt www.sidestep.pt A +norecursive

; <<>> DiG 9.2.2 <<>> @ns1.tvcabo.pt www.sidestep.pt A +norecursive
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10698
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 7, ADDITIONAL: 7

;; QUESTION SECTION:
;www.sidestep.pt.               IN      A

;; AUTHORITY SECTION:
pt.                     66823   IN      NS      ns.dns.pt.
pt.                     66823   IN      NS      ns2.dns.pt.

;; ADDITIONAL SECTION:
ns.dns.pt.              11611   IN      A       193.136.0.1
ns2.dns.pt.             7154    IN      A       193.136.2.226

;; Query time: 3 msec
;; SERVER: 212.113.161.227#53(ns1.tvcabo.pt)
;; WHEN: Fri Jan  4 17:31:59 2004
;; MSG SIZE  rcvd: 312
```

In this example, a query has been made to the DNS cache at ns1.tvcabo.pt for the www.sidestep.pt address (A) record. Since the query was made with the RD (recursion desired) flag not set and the record was not in its local cache, the server was unable to answer the query (note the ANSWER flag not set). Instead, it sent back information on authoritative servers for the .pt top level domain (TLD) that can be further questioned.

Note the following output, produced by the same command:

```
$ dig @ns1.tvcabo.pt www.sidestep.pt A +norecursive

; <<>> DiG 9.2.2 <<>> @ns1.tvcabo.pt www.sidestep.pt A +norecursive
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52370
;; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;www.sidestep.pt.               IN      A

;; ANSWER SECTION:
www.sidestep.pt.        28772   IN      A       193.126.14.52

;; Query time: 4 msec
;; SERVER: 212.113.161.227#53(ns1.tvcabo.pt)
;; WHEN: Sun Jan  4 18:35:22 2004
;; MSG SIZE  rcvd: 49
```

This time, the server returns a record that contains an answer to the query (the ANSWER section has entries). Since a non-recursive query has been made, there is certainty that the record was already cached locally, because a non-recursive query instructs the DNS cache not to use recursion in finding a response.

The Technique – The Polluting Way (Recursive Queries)

If only recursive queries are possible, there are still ways to determine with some degree of precision whether a given record is or is not present in the cache. There is, however, one major disadvantage: using recursive queries will pollute the cache, so if a given record is not present in the cache, it will be after the first query is made.

The output of a recursive query to a DNS cache follows:

```
$ dig @ns1.tvcabo.pt www.sidestep.pt A +recursive

; <<>> DiG 9.2.2 <<>> @ns1.tvcabo.pt www.sidestep.pt A +recursive
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44516
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.sidestep.pt.               IN      A

;; ANSWER SECTION:
www.sidestep.pt.        6458    IN      A       193.126.14.52

;; Query time: 13 msec
;; SERVER: 212.113.161.227#53(ns1.tvcabo.pt)
;; WHEN: Mon Jan  5 00:47:17 2004
;; MSG SIZE  rcvd: 49
```

There is a high possibility that query was already cached, as this can be confirmed in two ways. Firstly, the TTL (Time to Live) is suspiciously low; this is confirmed in the next output section:

```
$ dig @ns.sidestep.pt www.sidestep.pt A

; <<>> DiG 9.2.2 <<>> @ns.sidestep.pt www.sidestep.pt A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53517
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

;; QUESTION SECTION:
;www.sidestep.pt.               IN      A

;; ANSWER SECTION:
www.sidestep.pt.        86400   IN      A       193.126.14.52

;; AUTHORITY SECTION:
sidestep.pt.            259200  IN      NS      ns.sidestep.pt.
sidestep.pt.            259200  IN      NS      www.sidestep.pt.

;; ADDITIONAL SECTION:
ns.sidestep.pt.         86400   IN      A       193.126.14.51
```

```
;; Query time: 1 msec
;; SERVER: 193.126.14.51#53(ns.sidestep.pt)
;; WHEN: Mon Jan  5 00:59:06 2004
;; MSG SIZE  rcvd: 96
```

This query was made to the authoritative DNS server of the sidestep.pt domain and shows that the TTL of the cached response was much lower that the initial set TTL. This is a good indicator that the answer was already cached at ns1.tvcabo.pt.

Another way to look for cached responses is to observe the time that the query takes to process. If the query time is approximately equal to the round trip time (RTT) of a packet to the server, then the answer might have been already present in the cache. The author has learned from this technique on Rob Mayoff's email [4].

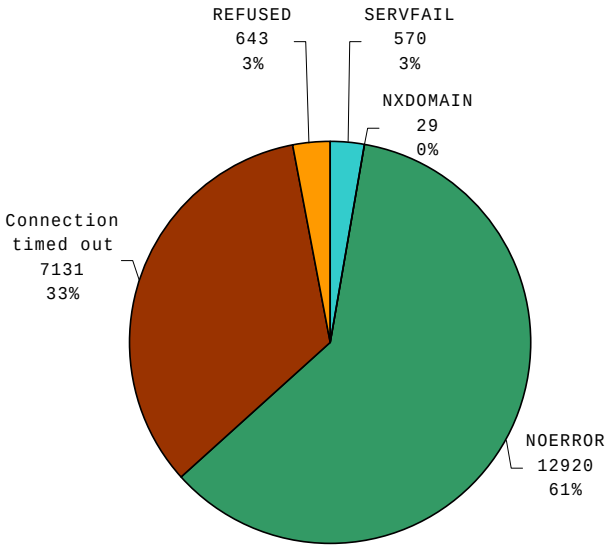
Note that the only tested DNS cache implementation that allows recursive queries while silently discarding non-recursive queries is dnscache, a software part of the djbdns package.

Gathering potential targets

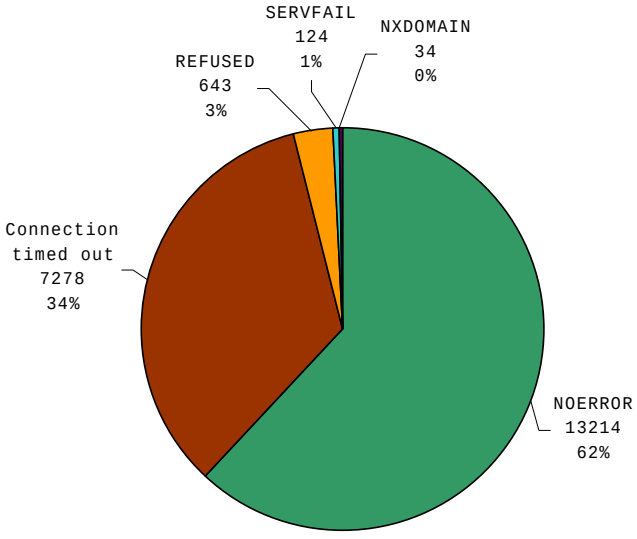
Before introducing ways to use the technique presented in this paper in an abusive way, the author of this paper will attempt to illustrate the current state of DNS server configurations over the Internet. A small empirical study was made to determine the extent to which it can be used around DNS systems worldwide.

To this, the author used the Internet DNS server list as compiled by Mike Schiffman in his recent DNS research report [5] and removed all private and non-routable IP addresses. For each server in the list, two UDP packets were sent, both containing a DNS query to translate the address www.google.com. One had the RD flag set, the other had not. All the results of this study, including updated versions of this paper can be found at http://community.sidestep.pt/~luis/DNS-Cache-Snooping/.

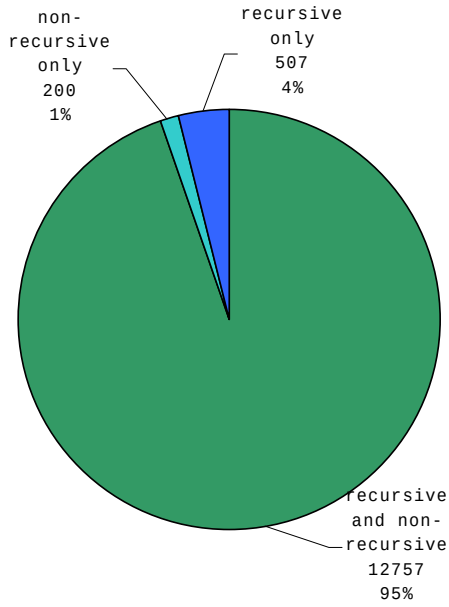
Graph 1:
Responses to Non Recursive Queries



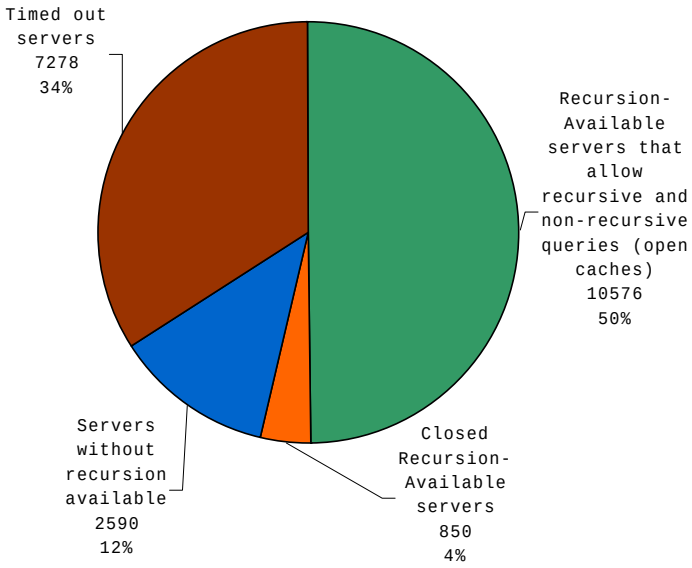
Graph 2:
Responses to Recursive Queries



Graph 3:
Distribution of NOERROR Responses



Graph 4:
"Open" DNS Caches



The results did not surprise. Of the 21293 systems queried, 13464 produced a good reply (NOERROR) on a non-recursive query to translate the address www.google.com (Graph 1). Out of those servers, 10576 had recursion available (RA flag set on the answer) and allowed recursive as well as non-recursive queries (Graph 4). These are "open" caches, as they can be used as caches and are misconfigured to allow snooping by everyone.

This means that approximately 50% of the servers queried (or 76% if only servers that produced a response are accounted for) are vulnerable to the DNS cache snooping technique. This means that if these servers are being used as a DNS cache by some given population, one can use the technique to extract information about its Internet surfing habits [6].

It appears that there is a target rich universe for using the cache snooping technique. The fact that there is such a high percentage of viable targets makes for even more interesting ways to abuse this issue.

Abuse Scenario #1 – A Typosquatters' Market Study

Typosquatting is a technique that became popular in early 2000, and involves putting up web sites with names that are common misspellings for popular sites such as Hotmail. For instance, the addresses www.hormail.com, www.hoymail.com and www.hotnail.com all redirect to www.superinternetdeals.com. Also, www.hotmial.com seems to be a search engine (concordantly, at the time of this writing, four of the ten most searched terms contained references to 'e-mail' or 'Hotmail'). Some well known cases of typosquatting include John Zuccarini [7], becoming known for registering more than 5000 domains, or the more recent (and drastic) attempt of Verisign's Site Finder of redirecting all non-existent .com domains to its search page [8].

The most obvious use for typosquatting is to generate revenue from advertising, but it can also be used to read email sent incorrectly for legitimate users [9] and even obtain possible active spam addresses.

Determining which domain generates more hits is usually guesswork. There is, however, a way to measure to some extent if a domain is worth registering. The following technique relies on the high percentage of DNS servers permitting non-recursive queries to all domains, as presented above.

The form of abuse is technically very simple: It involves querying a number of DNS servers that fulfill the following requisites:

- Cache records for a given network/population
- Answer non-recursive queries
- Cache no-such domain records (NXDOMAIN) [11]

The actual query is a non-recursive one that asks if the non-existent domain is present in the server's cache and if it is, then a typo as been made recently by the users of that cache.

As an example, the author of this paper, in his home network, attempted (unsuccessfully) to access the www.nonexistentdomain.net with his browser. Some time later, he snooped the local DNS cache with the following query, made on another machine on the same network:

```
$ dig www.nonexistentdomain.net +norecursive

; <<>> DiG 9.2.1 <<>> www.nonexistentdomain.net +norecursive
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 47913
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.nonexistentdomain.net.     IN      A

;; AUTHORITY SECTION:
net.                    10726   IN      SOA     a.gtld-servers.net. nstld.verisign-
grs.com. 2004020800 1800 900 604800 86400

;; Query time: 8 msec
;; SERVER: 192.168.11.21#53(192.168.11.21)
;; WHEN: Sun Feb  8 18:25:07 2004
;; MSG SIZE  rcvd: 116
```

As can be seen, the status of the reply is NXDOMAIN. The query is also cached, as can be observed by the decreasing TTL in the SOA line [11].

If another hostname is tested, for instance www.nottriedbefore.com:

```
$ dig www.nottriedbefore.com +norecursive

; <<>> DiG 9.2.1 <<>> www.nontriedbefore.com +norecursive
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3648
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 13

;; QUESTION SECTION:
;www.nottriedbefore.com.                IN      A

;; AUTHORITY SECTION:
com.                    170162  IN      NS      A.GTLD-SERVERS.NET.
<abbreviated>
com.                    170162  IN      NS      M.GTLD-SERVERS.NET.

;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET.     87409   IN      A       192.5.6.30
<abbreviated>
M.GTLD-SERVERS.NET.     31013   IN      A       192.55.83.30

;; Query time: 14 msec
;; SERVER: 192.168.11.21#53(192.168.11.21)
;; WHEN: Sun Feb  8 18:52:16 2004
;; MSG SIZE  rcvd: 472
```

As no attempt as been made to access this domain, it is not cached.

This can be automated to query a large number of "important" DNS caches (such as ISP caches). If the domain is present in a relatively high percentage of caches, it is a very good candidate for registering, as it will likely generate a significant amount of hits.

Abuse Scenario #2 – A Different Way of Measuring Site Accesses

Another interesting use of cache snooping is determining if the users of a given cache are accessing some domains more or less frequently. It can be useful to know if, for instance, a DNS cache used by a group of servers has the address entry www.windowsupdate.com in its cache (which probably means that at least some on the given population are being updated regularly), or if a certain obscure site is cached in multiple places.

The same technique can be used as with the previous abuse scenario, but now the search will be for valid cached records, instead of NXDOMAIN records. One can, for instance, automatically snoop a large number of DNS caches to verify how a certain site is being accessed around the world.

Note that all these measures are approximate. They loose accuracy in a measure proportional to the initial TTL of the Resource Record. For instance, if the initial TTL for the site www.sidestep.pt is 86400 seconds (24 hours) then this address will be present in a cache for that period and various accesses will only count as one. For better results it may be better to analyze the interval lengths in which a record is absent from a cache and then make an educated (statistical) guess about hit frequency.

Abuse Scenario #3 – Locating Users on the Internet

Consider that **Eve** is communicating remotely with **Bob** via some out of band medium, such as a telephone. Eve is also surfing the Web, but she doesn't want her IP to be revealed to Bob, so she is keeping herself on sites that seem trusted (trusteddomains.com) to avoid being logged. Bob, however, wants to locate Eve, so he tries the following:

1. Bob dares Eve over the telephone to go to http://evilhackers.go.here.trustedomains.com, which is a non-existent domain (NXDOMAIN).

2. After Eve unsuccessfully attempts to access, Bob will start to snoop a large number of reachable DNS caches for the NXDOMAIN resource record of the evilhackers.go.here.trustedomains.com domain.

3. If he finds a cached NXDOMAIN, there is a high probability that Eve is using that cache, which (also, probably) means she is located nearby (eg. same ISP).

Most DNS caches hold negative caches from 3 to 24 hours which gives Bob a while to look for the cached record. Given the vast number of DNS caches that are reachable from anywhere in the Internet, this attack scenario could be successful in some situations. But if Eve is warned, she can simply change her default DNS cache to point to a far away DNS cache, or to her own locally installed iterative resolver such as dnscache (inaccessible via the Internet).

Abuse Scenario #4 – Tracking Email Conversations

Sometimes it would be useful if there was a way to know if for instance, two companies are exchanging emails. Actually, it is possible to know if **Alice's** mail server has exchanged emails with **Bob's** mail server. All that our curious user **Eve** has to do is as follows:

First, Eve must obtain the IP of the DNS cache used by Alice's mail server to process queries. There are several easy ways to do this, one is described here:

1. Eve controls the DNS responsible for the eve.com domain

2. Eve sets up tcpdump on the authoritative DNS for eve.com to listen to MX queries for the evil.eve.com domain (non-existent)

3. Eve sends an email to a non-existent user at Alice's domain with a "from" address such as eve@evil.eve.com

4. Adam's mail server will attempt to bounce back an error message to eve@evil.eve.com and the MX query will be logged by tcpdump. The originating IP will be part of the DNS cache hierarchy used by Alice's mail server to attempt to find the MX record for evil.eve.com.

With the IP address acquired step 4, all Eve has to do is to snoop the DNS cache at that IP for the MX record of Bob's domain. If the record is cached, one can know that most likely a mail message has been transferred recently between Alice's and Bob's mail relays.

Note that this attack is only possible if Alice is using a reachable DNS cache. This is further discussed in the final section of this paper.

Abuse Scenario #5 – Time-based Session ID Generators

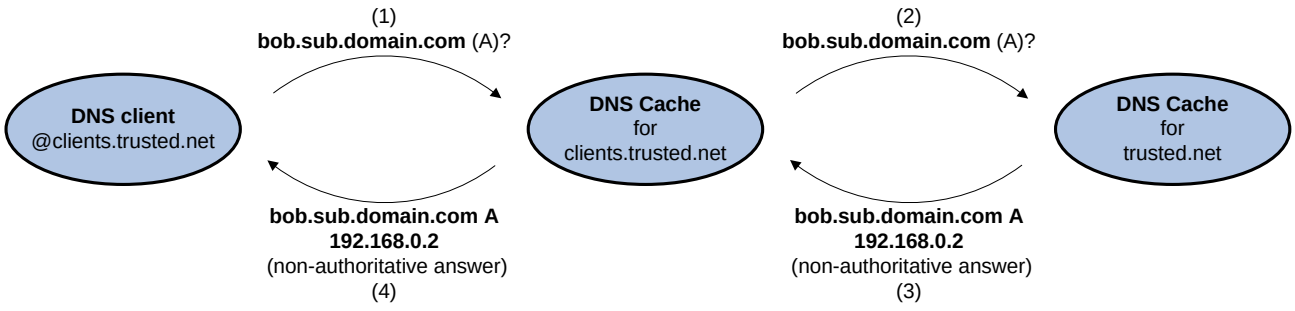
At every Web site that requires authentication, some form of maintaining session state between HTTP requests is required. Frequently, a session ID is generated when the user first accesses the site, and then that ID is passed to the user via a HTTP cookie, which is sent on every HTTP request. Some poorly written session ID generators rely solely on the actual time of day to generate its "pseudo-random" part. Usually seconds to microseconds are used for part of the entropy.

Thus, knowing exactly (by the second) when a user first accesses a given site can be helpful in predicting session ID's in an attempt to hijack a users session. Cached DNS information can help in this matter. By looking at a cached record's TTL, one can easily calculate the exact time when a user first accessed the time (the time at when the record was first cached) by subtracting the difference between the initial TTL set by the authoritative server and the cached TTL to the local time. This can be achieved by snooping the DNS cache that the user is using for name resolutions.

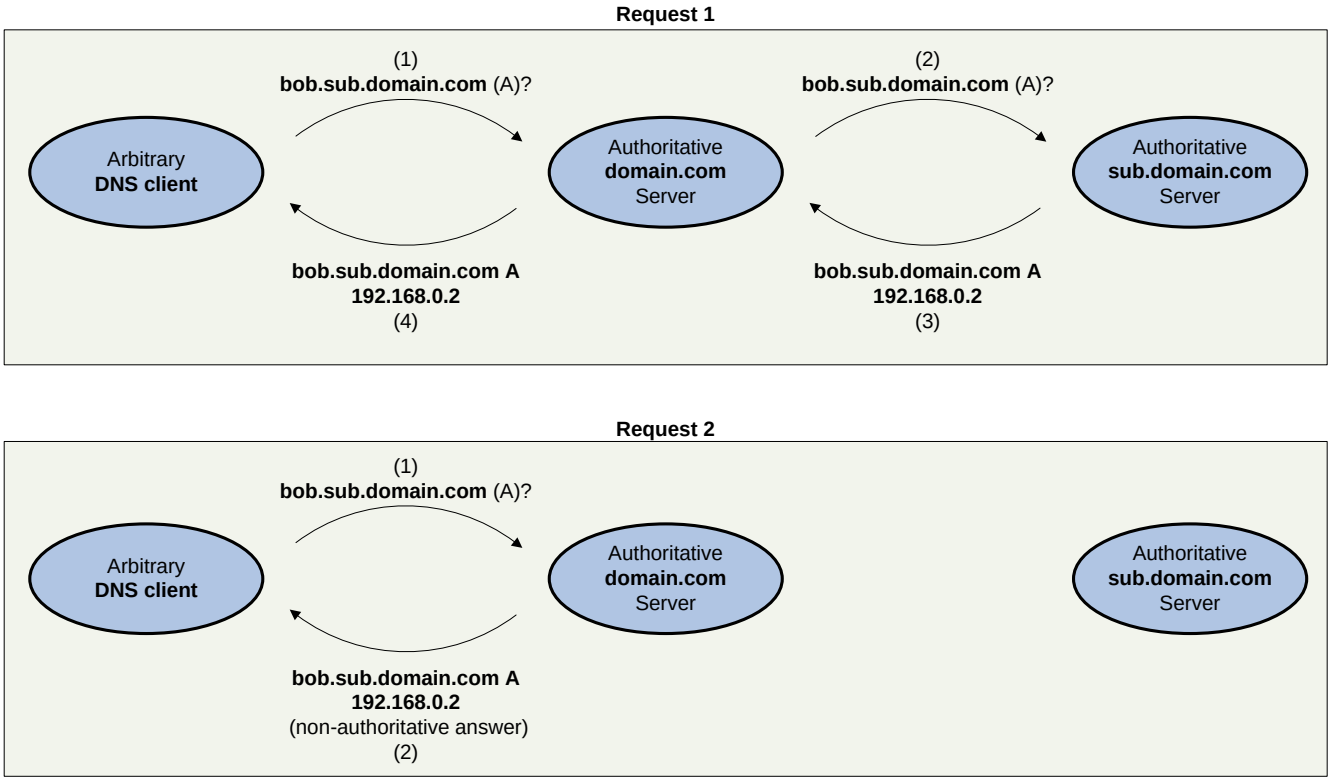
Recommendations for Reducing Exposure

There are several guidelines that are already available and significantly reduce the exposure to this problem.

Recommendation 1: firstly, DNS caches should only be allowed access by local users or child caches. The principle of caching is that of locality, so it makes no sense to allow a user from a totally different network to access your caches. The sheer number of "open" caches around the Internet facilitates abuse scenarios #1 to #3. As such, any DNS system (server/cache/hybrid) around the world should only respond non-authoritatively to known (and topologically close) clients. The following diagram illustrates two valid non-authoritatively DNS responses; in responses (3) and (4), there is an explicit trust relationship between endpoints.



Only one possible exception to the above guideline is considered. In the event of a DNS server/cache hybrid, it should be considered acceptable or even desired to cache information for child zones and to serve that information to the world. For instance, the authoritative server for 'domain.com' could maintain and serve cached records of child zones 'sub.domain.com' and 'abc.domain.com' to the world. For better understanding, consider the following diagrams.



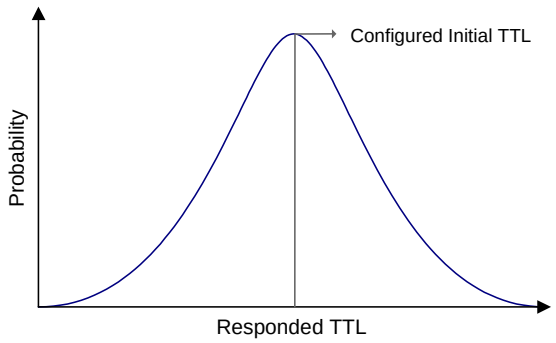
In request 1, the authoritative domain.com server caches the authoritative sub.domain.com server's response. In the subsequent requests such as request 2, it is allowed to respond non-authoritatively while this information remains cached. This can be useful to allow for temporary DNS server downtimes.

Recommendation 2: secondly, and most importantly, non-authoritative requests to DNS caches should not be allowed. For instance dnscache, a popular caching-only DNS implementation, tries to prevent cache snooping by refusing to answer non-recursive queries [3]. Another option is to never consult the cache when responding to non-RD queries.

Unfortunately, at the time of this writing, the ability to discard non-recursive queries to the cache while simultaneously allowing for non-recursive queries for the server's authoritative domains does not seem to be possible to do with BIND. An option for this configuration should be implemented on this software package in the future.

Also, despite being able to eliminate the possibility to do non-recursive snooping, it continues to be possible to snoop a cache using recursive queries, as described earlier in this paper. However, this method is much more error prone and disturbs the cache, so it might not be very useful to potential abusers.

Recommendation 3: there is also a third new recommendation to be made that can further reduce exposure to this risk. Some entropy can be added by DNS caches or even DNS servers when giving out the initial TTL for a given record.



When a DNS cache is about to add an entry to its cache, it could add some entropy to the TTL that it received from the authoritative server. By randomly adding or subtracting a small number of seconds to the TTL, DNS cache snooping becomes virtually impossible to do with only recursive queries. This measure also eliminates attacks such as those described in abuse scenario #5.

A Safer BIND Configuration

As a conclusion, it is interesting to note that while vulnerabilities in the BIND implementation can be blamed for the problems in the DNS infrastructure, poor configuration also seems to widespread. The following configuration [13] is to be taken as an example to a safer BIND configuration. These settings allow BIND to continue to be used as a cache by the networks "1.2.3.0/24" and "1.2.4.0/24", while still being able to respond authoritatively to queries regarding the domain "mydomain.com", while ignoring all others. Only relevant configuration options are displayed.

```
options {
        // to allow only specific hosts/networks to use the DNS server:
        allow-query { trusted; };

        // to allow only zone transfers to specific nameservers
        allow-transfer { other_ns; };
};

// Host / network grouping that maps “friendly” nameservers (such as secondary
nameservers)
acl other_ns {
        1.2.3.4;        // secondary 1
        1.2.3.5;        // secondary 2
        127.0.0.1;      // localhost
};

// Host / network grouping that maps networks that are able to do queries to other
// records besides the au
acl trusted {
        127.0.0.1;
        1.2.3.0/24;     // trusted net 1
        1.2.4.0/24;     // trusted net 2
};

// authoritative zone
zone "mydomain.com" in {
        type master;
        allow-query { any; }; // allow queries to be made to this zone by anyone
};

// reverse zone for the 1.2.3.0/24 network
zone "3.2.1.in-addr.arpa" in {
        type master;
        allow-query { any; }; // allow queries to be made to this zone by anyone
};
```

Endnotes

[1] For one side of the discussion of separating DNS caches and servers in different programs, read http://cr.yp.to/djbdns/separation.html.

[2] "dig" is a DNS diagnostic program that is part of the BIND software package, available at http://www.isc.org/sw/bind/

[3] dnscache information, including mentions to cache snooping and protective measures: http://cr.yp.to/djbdns/dnscache.html.

[4] Link to the original mail from Rob Mayoff concerning DNS cache snooping: http://article.gmane.org/gmane.network.djbdns/1002/

[5] "Bound by Tradition: A Sampling of the Security Posture of the Internet's DNS Servers" by Mike Schiffman, available at http://www.packetfactory.net/papers/DNS-posture/

[6] Note also that 62% (13214) of the inquired servers respond to arbitrary recursive queries to any host/domain. This means that a user can configure its Internet connection to use any of these DNS servers as a cache. This may not be desirable, as network resources are being used by others.

[7] "Domain Name Typosquatter Still Generating Millions", CircleID, http://www.circleid.com/article/101_0_1_0_C.

[8] "Verisign to Shut Down Site Finder", Wired News, http://www.wired.com/news/business/0,1367,60682,00.html.

[9] A search made in groups.google.com for emails ending in "@hotmial.com" returned almost 30,000 queries. This is indicative that a lot of people are erroneously typing their "from" or "to" address, thus inadvertently sending emails to the wrong domain.

[10] One empirical study was made in late 2000, when a user registered the domain jptmail.com, very little resembling hotmail.com. Even so, it managed to receive an average 32 email messages per day and totaled 3013 HTTP requests in one year. The results are available at http://www.nxdomain.net/jptmail.html.

[11] The caching of NXDOMAIN records was recommended mandatory by RFC 2308, available at ftp://ftp.is.co.za/rfc/rfc2308.txt.

[12] Brad Knowles has started a discussion by heavily criticizing djbdns. The relevant part is where he criticizes dnscache for not responding to queries with the RD bit cleared. The original mail to the freebsd-chat mailing list can be found here: http://groups.google.com/groups?selm=p05101519b8c51042d9db__10.0.1.8_%40ns.sol.net and the response from the author can be found here: http://cr.yp.to/djbdns/knowles.html.

[13] Note that even with this configuration, until the option that modifies the behaviour to non-recursive queries to non-authoritative domains exists in BIND, there is still the possibility for users in the "trusted" networks to snoop the cache of this server.