DNS Extensions Working Group
Internet-Draft
Intended status: Informational
Expires: April 2009

G. Barwood

October 26, 2008

Resolver side mitigations
draft-barwood-dnsext-fr-resolver-mitigations-08

Status of This Memo

Abstract

Describes mitigations against spoofing attacks on DNS, including:

(1) Repeating the query, including techniques for handling non-deterministic responses.

(2) Prepending a random nonce to the question where a referral is probable.

(3) Estimating the entropy available, taking into account
   (a) Observed packets with incorrect IDs.
   (b) The content of the cache.

Table of Contents

1. Introduction

This document describes mitigations that a resolver can currently deploy to resist spoofing attacks on DNS, without server software being updated.

The context in which these solutions were explored is CERT Vulnerability Note VU#800113, "Multiple DNS implementations vulnerable to cache poisoning".

The Kaminsky attack proceeds by asking a recursive DNS server a series of questions, each with a different random prefix, and then sending spoof packets to the server, containing additional records with genuine owner names but invalid data. For example:

Query:
Question <nonce>.com A

Spoof response:
Question <nonce>.com A
Authority: com NS ns.evil.com

The effect is to inject an invalid record into the cache.

Since the ID field in the DNS packet header is only 16 bits, a DNS server that does not deploy any mitigations can be compromised in a matter of seconds.

[An implementation of the techniques described can accessed at

http://www.george-barwood.pwp.blueyonder.co.uk/DnsServer/]

2. Criteria

These are resolver side solutions, thus only the resolver needs to be redeployed, or the software updated. This allows updated resolvers to be deployed immediately.

The solutions have to follow the DNS protocol.

The solutions have to be practical, non disruptive, and not anti-social.

3. Mitigations

Below, the resolver side mitigations are described.

Query repetition (3.1) is necessary and sufficient, the other mitigations reduce the number of queries needed for good security.

3.1. Query repetition

By repeating the query, additional entropy may be obtained.

Repetition is the only method of obtaining suitable entropy under all conditions, so a general purpose resolver MUST implement repetition.

A practical problem occurs when responses are non-deterministic, that is many different responses are obtained for the same question.

In this case, the resolver will need to perform an analysis to produce a converged result, or to report server failure (or a

security warning, if this is possible) if convergence has not been achieved after some iteration limit.

The suggested method is to accumulate entropy for various attributes of the response, specifically non-zero Rcodes (including an internal representation of no Data), the Resource Records (RRs), and the cardinality of each Resource Record Set (RRset).

Each Response can have a counter that represents the number of attributes that have not reached the required threshold. When the counter reaches zero, that response is considered fully checked, and is used as the converged result.

For example, suppose the question is MX records for example.com.

First response:
example.com MX mail1.example.com
example.com MX mail2.example.com

Second response:
example.com MX mail2.example.com (mail2.example.com confirmed)
example.com MX mail3.example.com

Also confirmed : example.com MX has 2 alternatives.

Third response:
example.com MX mail3.example.com (mail3.example.com confirmed)
example.com MX mail4.example.com

The result is the second response.

Note that it is possible for an attacker to break RRset integrity with a single forged response in the non-deterministic case. For example, the second response in the example could be forged. However this appears to be a very weak achievement.

Where convergence is very slow, some records may be omitted from the convergence test, and discarded (if not acceptable as described in section 3.6), to be fetched later as required.

The records that are always kept are

(E1) Records where the owner name and type exactly match the question.
(E2) NS records where the query question ends with the owner name.

Other records may be discarded (normally glue A records).

For example, if the question is www.example.com A, then in a response

www.example.com A 1.2.3.4 : is always kept by (E1)

example.com NS ns.example.com : is always kept by (E2)

ns.example.com A 1.2.3.4 : may be discarded

There is a possibility that combinations of resource records may result that would not occur normally. In the Akamai case, this could in principle result in a loss of resilience, instead of 9 distinct IP addresses for the name servers, some might be duplicated.

However no examples have yet been identified where a significant problem arises, and discarding records is only found to be necessary for the Akamai case, where full convergence might otherwise need about 100 queries. Stopping after about 10 queries typically results in one or two glue A records being discarded, and 9 NS records and the remaining 7 glue records being accepted.

In other cases, convergence generally occurs after at most 3 or 4 queries.

3.2. Randomize the case of the question (0x20)

Most authoritative servers preserve the case of the question in the response, so some additional entropy may usually be obtained by randomizing the case of the question.

3.3. Use a randomly chosen source port

This is a well-known method of obtaining extra entropy.

Unfortunately it is impractical for a program to reliably determine whether a resolver is currently situated behind a NAT device that may undo port randomization (and this can change for each packet sent), so a general purpose resolver MUST not rely on port randomization for security.

To avoid problems where authoritative servers may be behind firewalls that enforce very low limits on incoming UDP connections, resolvers MUST use the same source port when repeating a query (3.1).

3.4. Prepend a random nonce label to the question.

This msy be used where a referral is probable.

It allows an amount of entropy to be encoded limited only by the 256 character limit on a question, provided the authority server returns a copy of the question in the response.

If the response is not a referral*, the response should be discarded, and the query repeated without the nonce.

* That is any of the following are observed:
  (a) The response is Authoritative (AA bit is set in the header).
  (b) There is an error (RCODE is not zero).
  (c) The answer section is not empty.
  (d) The authority section is empty.

A simple heuristic for deciding where a referral is probable is:

(1) If the Bailiwick is Root or a TLD, and the question is not equal to the Bailiwick a referral is probable.

(2) Otherwise a referral is not probable.

3.5. Maintain a count of observed Bad IDs

The approximate number of incorrect IDs observed in some fixed time period, for example the last 20 seconds, may be kept.

This value may be used to decide when to deploy mitigations, such as extra query repetition, and allows a smooth response to attacks, while maximising performance under normal conditions where no attack is observed.

3.6. Use of calculated entropy

When a response is received, an entropy calculation may be performed to estimate how many bits have been checked.

It will typically include 16 bits for the ID, 0x20 bits, bits from the prepended nonce, and discount for unusual / non-standard features (such as IP mismatch, question not copied).

The entropy is accumulated for each response attribute, as described in 3.1, and a decision is then made to decide whether a value is to be accepted as valid, which in turn affects whether the query needs to be repeated as described in 3.1.

For example, the test for whether a value is valid could be

E + C > 50 + 2*K

where
- E is the accumulated entropy
- C is zero if the value is not in the cache, otherwise 30
- K is the logarithm (base 2) of the Bad Id count (3.5)

Cache entries may be retained in the cache for some period (say 1 day) after their normal TTL expiry time, to reduce the number of queries when the value needs to be refreshed after TTL expiry.

4. Analysis

This section is intended to be less formal, to give some insight into the rationale for the recommendations given in section 3, and to discuss possible adverse effects.

The intention is that these mitigations have minimal effects, other than to make DNS spoof attacks impractical.

4.1. Query repetition

Query repetition should have no impact other than on server load. Servers do not normally retain any state information about clients after the query/response transaction completes.

4.2. Impact on Root and TLD servers

The random nonce (3.4) is valuable because it means that no extra queries to Root and top level servers are needed in normal operation. This is important because these servers constitute the shared public base of the DNS, so the stability of these servers is very important.

The exceptions are the initial root "priming" query and queries for non-existent domains. For the root domain, by assuming that every child domain has an SOA record, Name Errors need not be retried (by checking the ower name for the SOA record). While this assumption is currently correct (and is also observed to be true for net and com domains), implementors need to carefully weigh any performance advantage with the risk that the assumption may not be valid in future.

Clients in general should implement user interfaces that make it

unlikely that users will enter invalid domain names, and that errors are properly notified, so they can be corrected. However this is outside the scope of this document.

In practice, most root server queries emanate from mis-configured software, so in any case proportional effect on root servers will be small. It is important that negative results be properly cached.

4.3. Impact on other levels

For the example test given in 3.6, two queries are usually required the first time a record is fetched. However when the TTL expires, the refresh operation only requires a single query.

It is expected that such refresh operations dominate proper DNS traffic, so the impact should be minimal.

Operators of authoritative servers have several options if the query repetition may cause overload.

(a) Increase unreasonably low TTLs.
(b) Use names with more alpha characters (to take advantage of 0x20).
(c) Implement support for the proposed AL record or equivalent.

The latter implies that agreeing a specification for the proposed AL record type (or EDNS Ping equivalent) would be useful.

4.4 Lame servers and the random nonce

In order to resolve domain names where servers are incorrectly

configured, it may be necessary to use a query without the nonce.

A current example is resolving the IP addresses for the name servers for www.iahc.org, which are ns2.ar.com and ns3.ar.com.

The com nameservers generate a referral for the question <nonce>.ns2.ar.com, which leads only to lame name servers, but the IP address for a non-lame server when the nonce is omitted.

Thus when lame servers are detected, special logic to allow name resolution to still occur is needed.

Of course a resolver may choose to merely report failure in this case, however this may not be practical.

4.5. Security Level

The 50 bits suggested in 3.6 should provide a good margin of safety. An attack sending one spoof packet every 20 seconds at a particular target will take about 50 million years to succeed.

Taking Bad IDs into consideration (3.5) implies that an attacker gains nothing from sending attacks at a faster rate.

As a test, the resolver was run with the security level set to 200 bits with no perceptible decrease in performance (the required number of packets can be calculated in advance and sent in parallel, except in the non-deterministic case).

5. Security Considerations

All of the mitigations aim to provide more security. Query repetition has an obvious adverse effect on performance and bandwith.

Each query repetition provides an extra attack opportunity, so the total entropy requirement may be adjusted to reflect this.

The random nonce may expose internal state to an attacker who controls a name server. It is essential that a cryptographically strong source of random numbers be used to generate IDs, 0x20 bits

and prepended nonces. This must be seeded from data that cannot be guessed by an attacker, such as thermal noise or other random physical fluctuations.

6. IANA Considerations

No direct considerations.
Indirectly, the TYPE code for AL record described in 4.4.

7. Acknowledgments

Thanks to Nicholas Weaver (ICSI Berkeley) and Wouter Wijngaards (NLnet Labs). The idea of prepending a nonce may be due to Paul Vixie (ISC).

8. Informative References

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

Author's Address

George Barwood
33 Sandpiper Close
Gloucester
GL2 4LZ
United Kingdom

Phone: +44 452 722670
EMail: george.barwood@blueyonder.co.uk
Skype: george.barwood

Intellectual Property