What's in a Name (Collision)? Modeling and Quantifying Collision Potential

Casey Deccio
Verisign Labs

ABSTRACT

Name resolution processes and the presence of parallel namespaces outside the IANA-rooted DNS create the possibility for the collision of names, resulting in some level of degraded functionality, security, or privacy. Previous work has produced various quantitative analyses based on observations of Internet traffic and data, including DNS queries and Web content. From these studies, behavior and associated risks have been inferred. However, we suspect that it is difficult to draw conclusive results from these studies performed "from the outside in" because there are so many unknowns. In this paper, we reverse the process: we start with a fundamental model of name resolution at a client, use this model to define risk, and use data to support the model. With this foundation, we identify methodology to apply the model to quantify and mitigate risk given the configuration of a network or system.

1. INTRODUCTION

The translation of names to resources (e.g., Internet Protocol (IP) addresses) is a fundamental service, necessary for proper functionality of most Internet applications and general use by human consumers. The burden of this mapping falls mostly on the Domain Name System (DNS) [1, 2], a hierarchical directory of domain names, globally available and distributed.

While the public DNS typically results in consistent and correct responses, the name resolution process is more complex than a simple query to the public DNS. Given certain configurations, behaviors, and varying views of the DNS between public and private environments, there is potential for collision of names, which might cause incidental or intentional name resolution failures. The impact of such collisions might range from trivial to catastrophic.

Much previous work has been conducted in this area, early on in response to buggy resolver behavior [4], and more recently in conjunction with the 2013–2014 introduction of new generic top-level domains (gTLDs) into the DNS[8, 5]. In this paper, we explore the concept of name collisions by formalizing a model of name resolution, as currently specified and implemented. We use this model to formally define name collisions, quantify their impact and risk, and identify methodology to apply the model in production DNS environments.

The primary contributions of this paper are:

- a general model for name resolution at the system level;
- definitions and metrics to quantify the risk associated with name collisions; and
- applied methodology for the detecting and alerting of name collision risk.

We expect that the model and applied methodology laid out in this paper can help organizations, both local and global, to identify, quantify, and mitigate the potential for name collisions and their consequences.

The remainder of this paper is organized as follows. We summarize previous work in Section 2. In Section 3 we detail the mechanics behind name resolution at the resolver library of a host. We formalize these mechanics into a model and define collisions within the context of that model in Section 4. We propose a process applying the model and aggregating the results to compile a global view of name collision potential in Section 5. In Section 6 we conclude our work and include future work in fleshing out the model.

2. PREVIOUS WORK

Several previous studies have assessed name collisions in the DNS. One of the earliest reports was RFC 1535 [4], which identified problems with buggy DNS resolver implementations and their mishandling of suffix search lists, resulting in potential name collisions. Several recommendations were included in the RFC for improving name resolution accuracy and consistency.

With the 2013–2014 introduction of new gTLDS into the DNS, several large-scale studies have been performed to study many aspects of collisions and their risks. A study commissioned by the Internet Corporation for Assigned Names and Numbers (ICANN) looked at data from queries received at root and other DNS servers to assess the risk associated with delegating names (i.e.,

potential TLDs) that have been applied for based on the patterns identified with the queries observed [5]. Another report [8] produced similar analysis, including additional techniques for quantifying risk, based not only on DNS requests, but also World-wide Web content, X.509 internal certificates, regional preferences, and query timing metrics.

In this paper, we complement previous studies with an analysis from the perspective of the client, rather than the Internet server. The model presented in this paper can be applied to further validate the data-centric studies previously presented.

3. NAME RESOLUTION

In order to understand name collisions in the DNS, their causes, and their potential impact, we present a model of name resolution from the standpoint of an application using the resolver library of a modern operating system (OS). We first examine the process surrounding resolution of a name provided to the resolver library. When a name is presented to the resolver library, the process for its resolution varies depending on resolver policy specific to its implementation and related configuration.

3.1 Search Suffix List Processing

The name resolution process typically results in one or more DNS queries being made to a recursive DNS resolver, the final response from which determines the end answer returned by the resolver library. What name(s) are queried, however, depends on suffix search list processing [3]—that is, applying a series of suffixes to a name that is otherwise considered not "fully qualified" by the library. There are several representative behavioral outcomes exhibited, identified in previous research[6] and confirmed in our study:

- *never*. The name is always treated as a fully qualified domain name (FQDN) and is exclusively queried of the recursive DNS resolver as is.
- *always*. The name is never treated as an FQDN, and the search list is always applied to the name to form queries for the recursive DNS resolver.
- *pre*. The name is first treated as unqualified, such that the search list is applied; finding no positive response (i.e., no data correponding to the name) the name is queried of the recursive DNS resolver as is.
- *post*. The name is first treated as an FQDN, first being queried of the recursive DNS resolver as is; finding no positive response, the search list is applied to the name, resulting in additional names queried of the recursive DNS resolver.

OS	1 label	> 1 labels
Windows XP	always	post
Windows 7/8	always	never
Mac OS X (10.9.1)	always	never
Debian GNU/Linux 7	pre	post

Table 1: A summary of search order behaviors for popular modern OSes for single- and multi-label names not ending in "."

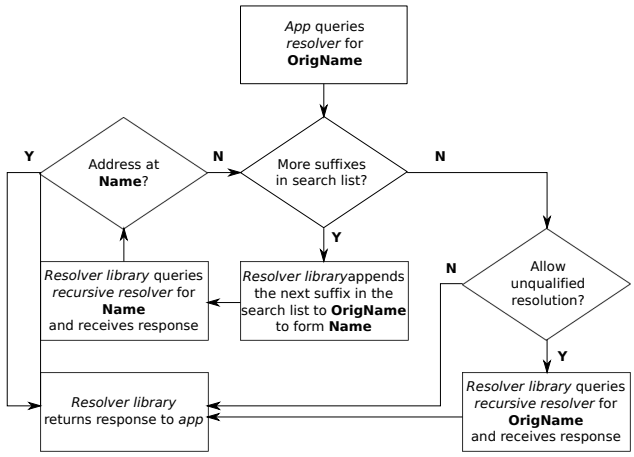

Figure 1: Resolution flow for an unqualified single-label name provided to a resolver library.

There are two primary considerations for whether or not a name is considered "fully qualified". If the name is explicitly denoted as a rooted name, indicated by its termination with a ".", then resolver libraries of all OSes consistently treat the name as an FQDN, and the search list is never applied. However, this notation is typically not used by users and applications, and lacking this marking, behavior of resolver libraries varies across OSes, depending on whether the queried name has one or more *labels* (i.e., "words" between the dots in a DNS name). Our findings are consistent with previous research [6] and are summarized in Table 1[1], and the name resolution flow for such single- and multi-label names is captured in Figures 1 and 2, respectively.

3.2 Name Resolution Realms

Computer systems, while often mobile, are typically configured for a "home" environment. For corporate systems, this might be their internal corporate network; for a university, it might represent the university network. While hosts based in a "home" environment

[1]Note that some Linux *applications* take exception to this behavior, overriding or supplementing OS resolver library behavior with their own. In particular, the `postfix`, `exim4`, `sendmail`, and `qmail` mail transfer agents all have distinct behaviors with respect to search suffix list processing.

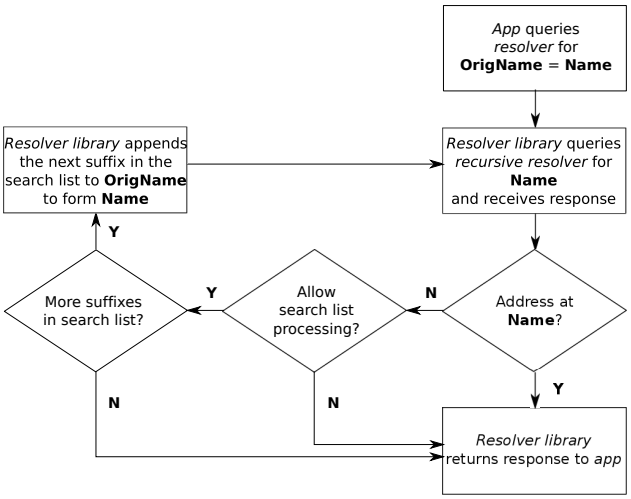

Figure 2: Resolution flow for an unqualified multi-label name provided to a resolver library.

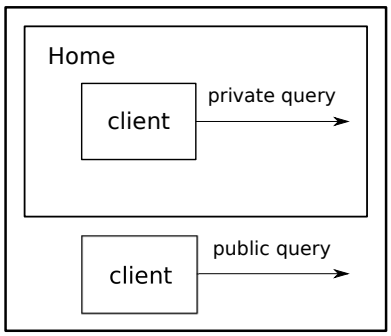

Figure 3: Private queries originating from the home realm, and public originating from outside the home realm.

might have different OSes and thus resolution behaviors, the suffix search list (if any) is often consistent for systems within a common environment. Additionally, it is not uncommon for home environments to host private versions of DNS namespace, in which names might resolve differently when queried from home than they would from the public Internet.

Note that that when we refer to the DNS namespace reachable from from the public Internet, we are referring to the DNS rooted at the Internet Assigned Numbers Authority (IANA)[2]. Thus "public DNS servers" refer in this context to servers delegated (one or more levels deep) from the IANA-rooted DNS.

It is possible that some systems are configured for more than one home environment, including different sets of search lists. In this paper we assume a single home environment, though the concepts might be expanded to consider multiple home environments.

Throughout this document we refer to two perspectives of DNS resolution with respect to a system. *Private* and *public* resolution refer to queries made from the home realm and those made from the the public realm, respectively. Such perspectives are illustrated in Figure 3. Note that a private query doesn't mean that the queried name is hosted privately and will be answered from authoritative sources in the home environment. Rather, it means that such *could* be answered by local authoritative sources, if such exist in the resolution path from the home environment. Public queries, on the other hand, always reach public DNS servers.

4. NAME COLLISION MODEL

In this section we model name resolution by defining relevant terms and applying them to metrics involving name collision potential and risk.

4.1 Terminology and Notation

Using the processes described in Section 3, we define terms related to resolution to reference in the creation of our model.

Resolver behavior.

Let C denote the set of resolver behaviors for OS resolver libraries. Each behavior $c \in C$ corresponds to policy that is followed to determine whether queried names are already fully qualified or require additional suffixing, following the process outlined in Section 3. Each row in Table 1 represents a behavior, $c \in C$, for the OS listed.

Suffix search list.

Let $S = [s_1, s_2, \ldots, s_p]$ denote the sequence of suffixes comprising the suffix search list configured for use by the resolver library on a given system.

[2]See https://www.iana.org/.

DNS answer.

For a name, n, queried of a DNS resolver, let $A(n)$ denote the answer (positive or negative) as returned by the public DNS, and let $A'(n)$ denote the DNS answer returned in the private environment.

Existence of DNS name and data.

Let boolean values $YX(n)$ and $YX'(n)$ denote the existence of name n in the public DNS and as returned in the private environment, respectively. Similarly, let the values $YA(n)$ and $YA'(n)$ denote the respective public or private existence of DNS data (e.g., IP address) corresponding to n [3]. If $YA(n)$ is true, then the public response is considered *positive*; otherwise it is considered *negative*. The same is true for private responses. Note that because $YA(n) \rightarrow YX(n)$, if $YX(n)$ is not true, then the response is also considered negative.

DNS names recursively queried.

Given a domain name, n, provided to an OS resolver library having resolution behavior $c \in C$ and configured with suffix search list S, the sequence of names queried by the resolver library necessary to produce the intended resolution result of n in the private environment is:

$$Q_{(c,S)}(n) = [n_1, n_2, \ldots, n_m]$$

such that the desired DNS response for names $[n_1, n_2, \ldots, n_{m-1}]$ is always negative and $A'(n_m)$ is the intended answer for n in the private environment. The private response, $A'(n_m)$, may be either positive or negative; the value of $A'(n_m)$ depends on whether or not a positive answer is anticipated.

Publicly delegated to organization.

Let boolean value $D(n)$ reflect that the querying organization is under delegated administrative control of n's namespace in the public DNS.

Locally administered.

Let LA denote the set of DNS domains locally administered in a system's home environment, such that private (home-based) queries for names that are under any of these domains are answered from local authorities and do not reach the public DNS.

Ancestral names.

For a given name, n, queried of a recursive DNS resolver, the sequence of ancestral names, beginning with the highest level (e.g., `example` before `foo.example`) is denoted:

$$L_n = [l_1, l_2, \ldots, l_q]$$

[3]While the most common application of the DNS is the mapping of names to IP addresses, addresses are simply one type of mapping. We use the term "data" because the model is agnostic of type.

Locality.

Let h denote the percentage of time in which systems are local to their home DNS resolution, i.e., in which DNS queries for names in locally-administered namespaces will resolve without reaching public DNS.

4.2 Definitions

The *collision* of a name queried of a resolver library involves the co-existence, co-incidence, and conflict of namespace between private and public DNS during any part of the name resolution process for the name, i.e., for any names generated for DNS queries. Either (or both) of the two following characteristics indicate a collision for name n:

- There is at least one name, $n_i \in Q_{(c,S)}(n)$, such that the namespace for n_i is locally administered, but a response to a DNS query for n_i (public or private) is received from the public DNS.

- There is at least one name, $n_i \in [n_1, n_2, \ldots, n_{m-j}]$ (i.e., the first n_{m-j} names in $Q_{(c,S)}(n)$), $j \in [0, 1]$, such that the response to a DNS query for n_i (public or private) results in a positive response. If the expected response for n_m is positive, then j is 1; otherwise j is 0.

We define two specific classes of name collisions: *active* and *passive* collision. In an active collision, the final resolution result for n differs from the intended result, either because one of $[n_1, n_2, \ldots, n_{m-j}]$ resulted in a positive response or because the final response received from the public DNS differed from the anticipated response from a local DNS source. In a passive collision, the final result is equivalent to the anticipated result.

While both active and passive collisions have some impact, the effects are different, which is why we differentiate them in this paper. With passive collisions the user/application experience is no different than if the collision had not occurred, but the fact that queries unintentionally reach authoritative servers in the public DNS allows possible surveillance by third parties, yielding privacy concerns. With active collisions, the user/application is interrupted, in that the response received is different than what they expected. This might have more serious functional, privacy, or security implications.

Collision probability, $0 \leq P_{(c,S)}(n) \leq 1$, is the likelihood that resolution of name n on a system with resolver behavior c and suffix search list S results in collision. It is defined in collisionProbability.

In the simplest sense a name collision might not involve leaving organizational boundaries. Such a collision is certainly the result of misconfiguration, and there would be ill effect, but it is perhaps more a nuissance than an issue of privacy or security.

```
if ∃n_i ∈ Q_(c,S) (n) | n_i ∈ LA then
  localAnsFromPub ← 1 − h
else
  localAnsFromPub ← 0
end
falsePosProb ← 0 ;
if ∃n_i ∈ [n_1, n_2, ..., n_{m−j}] | YA′(n_i) then
  falsePosProb ← falsePosProb + h
end
if ∃n_i ∈ [n_1, n_2, ..., n_{m−j}] | YA(n_i) then
  falsePosProb ← falsePosProb + (1 − h)
end
return max[localAnsFromPub, falsePosProb]
```

Function collisionProbability(n)

A more severe case is one in which names intended for local resolution are answered by third parties. *Third-party false positive risk* of a DNS name quantifies the potential risk associated with receiving a positive response from a public DNS domain delegated to a third party for a name for which a negative response is anticipated. It is described in thirdPartyFalsePosRisk.

```
if D(n) then
  return 0
end
if YA(n) then
  if n ∈ LA then
    return 1 − h
  else
    return 1
  end
end
risk ← 0 ;
foreach l_i ∈ L_n do
  if YX(l_i) then
    if l_i ∈ LA then
      risk ← risk + (1 − h) × 0.5/i
    else
      risk ← risk + 0.5/i
    end
  end
end
return risk
```

Function thirdPartyFalsePosRisk(n)

```
if D(n) then
  return 0
end
if n ∉ LA then
  return 0
end
return 1 − h
```

Function thirdPartyLeakageRisk(n)

If the name is delegated to the querying organization in the public DNS, then the third party false positive risk is always 0, even if collision is possible. Otherwise, if the public DNS response is positive (i.e., $YA(n)$ is true), then the result depends on whether a version of the domain is also locally administered. If it is, then the risk is dependent on locality, h; otherwise, the risk is 1, indicating that collision is certain.

While a given name, n, might not exist in the public DNS, some of its labels might be delegated in the DNS to entities other than the querying organization. If there is no positive response for the name itself, then risk is calculated by adding the logarithmically decreasing values for each ancestral name, $l_i \in L_n$. The rationale for this is that there is a big jump in potential for collision when the name goes from "non-existent" to "delegated" at the root level. Thereafter, for each label (often representing organizational boundaries) that is further delegated, the cumulative value of risk is increased by half the value of the previous increase, asymptotically approaching 1. These incremental values are weighted according to locality, h, if they are locally administered.

Risk of leakage to third-party delegations of a DNS name is a similar metric that quantifies the potential risk associated with getting a DNS response from a third party when such is expected to come from local sources. It is defined in thirdPartyLeakageRisk.

Third-party collision risk of a name passed to a resolver library can be quantified using third-party false positive risk and third-party leakage risk as independent probabilities. It is quantified by applying the other two metrics on the DNS names iteratively queried for n, $Q_{(c,S)}(n)$. We also introduce $I(n)$, $0 \leq I(n) \leq 1$, to denote the organizational importance of the name n as an additional factor towards the risk metric. The value of $I(n)$ might be set based on the number of queries it sees or the criticality of the service it provides, for example. The thirdPartyFalsePosRisk algorithm is evaluated for names $[n_1, n_2, \ldots, n_{m-1}]$ and aggregated with thirdPartyLeakageRisk for n_m. The resulting algorithm is shown in thirdPartyCollisionRisk.

4.3 Case Studies

We apply some case studies to understand the implications of the model described in earlier parts of this section.

Let us first start with a scenario in which there are no suffixes in a search list and there is no locally served namespace, i.e., $S = []$ and $LA = \emptyset$. If a client application in the organization queries for the name `foo.example`, then there is no chance of collision, i.e., collisionProbability returns 0. It is thus also true that there is no third party delegation risk. Note that this is independent of whether or not the name or its parent is delegated (at the time of this writing they are not) and whether or not a positive or negative response is anticipated.

Now, we assume that the organization employs a search list with just a single suffix, `foo.example`, and a query is made by an application for `www`, intending to reach `www.foo.example`, which should have a positive result. The list of query names, $Q_{(c,S)}(\texttt{www})$, is simply:

[`www.foo.example`]

return $I(n) \times \left(1 - \left(\prod_{n_i \in [n_1, n_2, \ldots, n_{m-j}]} 1 - \texttt{thirdPartyFalsePosRisk}(n_i)\right) \times (1 - \texttt{thirdPartyLeakageRisk}(n_m))\right)$

Function thirdPartyCollisionRisk(n)

and the collision probability and risk are again 0. However, when `foo.example` is delegated to a third party in the public DNS and is also locally administered by the organization, then the collision probability and risk are both subject to locality, h. Specifically, both are $1 - h$ (note that we assume, importance, $I(n)$ to be 1 throughout this paper).

With search list and locally administered namespace continuing unchanged, if it is now instead expected that the response for `www` is negative, then there are several additional considerations. The list of query names, $Q_{(c,S)}(\texttt{www})$, remains the same for some configurations, c, and for such the collision probability and risk remain the same, $1 - h$. However, for configurations, c, that try the single-label name after applying the search suffix (i.e., *pre*), the list of query names, $Q_{(c,S)}(\texttt{www})$, becomes:

$$[\texttt{www.foo.example}, \texttt{www}]$$

For systems with such configurations the collision probability and risk are both still $1 - h$, as long as the top-level domain `www` does not exist (it does not exist at the time of this writing). However, if we say that delegation exists (but does not resolve to any addresss), then the collision probability becomes 1. The collision risk associated with third-party delegation becomes:

$$1 - (1 - (1 - h))(1 - 0.5)(1 - h) = 1 - 0.5h^2$$

While resolution of "dotless" (i.e., single-label) domains to data is strongly discouraged, and even prohibited for new generic gTLDs[7, 9], we assume simply to continue our example that `www` resolves to an IP address. In this case, the collision risk associated with third-party delegation becomes 1.

5. MODEL APPLICATION

The model presented in Section 4 is only a piece to the greater puzzle of name collision potential and risk. The metrics introduced apply to a given name on a given host system with a given configuration. Some of the components from which the metrics are derived are deterministic, such as the search list and the configuration behavior. Others are variable, like locality and importance factor.

To apply the collision model to larger set of clients, representative values must be set for the components, and the metrics must be aggregated. Determining appropriate values requires an understanding of client configuration in the environment, supplemented with measurement and monitoring using careful instrumentation within home environments and without.

The primary base values needed are the suffix search list with which clients are configured and the locally administered DNS namespaces. This can be determined by administrators familiar with the configuration of client systems deployed for the home environment of the organization. While it is possible that various search list configurations exist for the environment, in this paper we assume a single purpose and leave expansion of this model as a reasonable contribution out of scope of the current work.

Monitoring at the DNS recursive resolvers pointed to by clients in the home environment is key to learning some of the other values from which collision metrics are derived. By observing queries at the recursive resolver and knowing the suffix search lists, the names queried to the resolver libraries on the clients can be deduced, their frequency, and often the distribution of resolver configurations (e.g., *always*, *pre*, etc.). Query frequency can be used to determine an appropriate per-name importance value.

The per-organization locality value, h, is arbitrary but can be set based on actual measured data and/or modified to assess different scenarios. Likewise, the private or public existence of certain names can be gleaned from passive observation or from active probes, but can be modified to introduce potential scenarios, such as the delegation of new gTLDs or SLDs.

A more detailed procedure for setting and gleaning these variables associated with quantifying name collision probability and risk will be part of future work, but the principles described in this paper provide a reasonable foundation for such application of the model.

6. CONCLUSION

In this paper we have broken down the process of name resolution at the resolver library of a client system and formalized a model from which we have derived metrics to define and quantify associated risk. Additionally, we have introduced techniques for applying the model in an network environment to obtain a reasonably accurate understanding of collision potential and risk.

We expect with the application of this model to more accurately describe the impact of name collisions both presently and in conjunction with network changes, such as additional mobility, new OS behaviors, or the delegation of new gTLDs. Data from previous and future studies, based on DNS queries at authoritative servers and other sources, can supplement and corroborate this model.

7. REFERENCES

[1] RFC 1034: Domain names - concepts and facilities, Nov 1987.

[2] RFC 1035: Domain names - implementation and specification, Nov 1987.

[3] RFC 1123: Requirements for internet hosts – application and support, Oct 1989.

[4] E. Gavron. RFC 1535: A security problem and proposed correction with widely deployed DNS software, Oct 1993.

[5] Interisle Consulting Group. Name collision in the DNS, Aug 2013. https://www.icann.org/en/about/staff/security/ssr/name-collision-02aug13-en.pdf.

[6] Geoff Huston. New gTLD concerns: Dotless names and name collisions, Nov 2013. https://labs.ripe.net/Members/gih/dotless-names.

[7] ICANN. New gTLD dotless domain names prohibited, Aug 2013. http://www.icann.org/en/news/announcements/announcement-30aug13-en.htm.

[8] Verisign Labs. New gTLD security, stability, resiliency update: Exploratory consumer impact analysis. Technical Report 1130008 version 1, 2013.

[9] ICANN Security and Stability Advisory Committee. SSAC report on dotless domains, Feb 2012. http://www.icann.org/en/groups/ssac/documents/sac-053-en.pdf.