Whitepaper

Dealing With Short Search Terms When Using DNSDB Flexible Search

Joe St Sauver

DOMAINTOOLS®

Table of Contents

Introduction & Caveats

DNSDB Flexible Search lets you conduct powerful searches of DNSDB RRnames (or DNSDB Rdata) for simple keywords or arbitrary "regular expressions." Flexible Search is bundled with your DNSDB subscription at no additional charge and acts as a DNSDB "finding aid," complementing and enhancing DNSDB Standard Search, allowing users to search DNSDB for patterns they previously couldn't find directly.

For many normal (reasonable-length/complexity) search strings (especially company names or trademarks such as "Airbus", "Boeing", "Embraer", etc), finding matching names in DNSDB is a straight-forward task that will return a moderate number of hits.

The Problem of Short/Common Search Strings: Sometimes, however, you may have a keyword or brand that may only be a few characters long (or which is an extremely common word). When that's the case, you may run into LOTS of results that technically "match," but which match due to effectively random content which won't be of substantive interest to a brand manager or a company's cybersecurity team. This can trigger a more complex analytical process. The process gets complicated because you may be running into randomized "wildcard" domains or other DNS "noise."

Alternatives to Machine Learning: Some might turn to machine learning (ML) approaches to cope with randomized names (see for example https://www.farsightsecurity.com/blog/txt-record/randomdomains-20190709/ and https://www.farsightsecurity.com/blog/txt-record/nvidia-20200124/), but this article illustrates a more interactive/"craft"/analyst-driven approach to that challenge, for those whom might want to avoid ML-based approaches for whatever reason.

CAUTION: For your own safety, we urge you to REFRAIN from going to (or otherwise directly engaging with) sites you happen to see in passive DNS results (including passive DNS results mentioned in this report). This can be a risky way to try to assess those sites. Some such sites may attempt to drop malware on your system, or negatively respond to unwanted attention with DDoS attack traffic. Only visit sites you've uncovered if you're a trained cybersecurity professional and can safely do so from an easily-reimaged lab system, and knowingly accept the risks associated with doing so.

Preventing Accidental Visits to Risky Domains: To prevent accidental filtering of this report, and to avoid accidental contact with potentially risky domains, we'll normally show domain names in this document in one of two alternative formats:

- "Defanged" format (where one or more "real dots" in the domain name has been replaced with [dot])
- Label-reversed format (so that www.example.com would be shown as com.example.www instead)

When you're actually using one of these domain names in a command, you'll need to use the name in "normal" (non-defanged, non-label-reversed format).

There Is No Single "Right Way": The approach shown in this document is only ONE approach that some people may use. Others may find the approach described in this document to be entirely too reliant on professional judgment, or to be flawed for being willing to deprioritize or even discard some discovered domains. If so, that's fine — if something else works better for you, do that instead. We want you to use whatever works best for you. The purpose of this report is to illustrate one approach of many.

This Discussion Assumes a Un*x Command-Line Based Environment: We know that some of you may not be interested in (or comfortable) working at the Un*x command prompt under any circumstances. For example, perhaps you're on a Windows workstation, or prefer a GUI point-and-click environment to a Un*x command line environment.

Unfortunately, when tackling difficult short string/common string analyses, Un*x may be an environment that's hard to avoid. GUI environments often turn into "snowballs" when tackling data sets with millions of results. We hope this writeup will at least show you some of what's possible at the command line, and potentially tempt you into trying a Un*x environment for your professional work (that environment is lurking inside every Apple Mac, for example, so you may already be using a Un*x system without even knowing it — you just need to go to Applications --> Utilities --> Terminal to get to the shell prompt).

Phase I: DNSDB Flexible Search Phase

2. Finding Domain Names Containing the String "ibm"

An example of an iconic short string is "ibm", one of the trade names of International Business Machines. You might think "ibm" is a relatively uncommon string, but when you're searching millions of records, it can actually appear surprisingly often.

To verify this, let's search for the string "ibm" using the dnsdbflex command line Flexible Search client (see <https://github.com/farsightsec/dnsdbflex>).

Our first basic Flexible Search query returns over a million results (the maximum results we can request via any single query):

```
$ dnsdbflex --regex "ibm" -l0 -j > ibm-hits.jsonl
$ wc -l ibm-hits.jsonl
1042013 ibm-hits.jsonl
```

"Decoding" the above query (for those who may be "playing along from home"):

dnsdbflex	This is our command line DNSDB Flexible Search client
--regex "ibm"	The regular expression we want to match, in this case, the string ibm
-l0	Return the maximum number of available hits for our query
-j	Return those hits in JSON Lines format

> filename Redirect the output from this command to the specified filename

We can augment our million-plus initial results by requesting three additional "tranches" or slices of results (each representing roughly another million results) by saying:

$ **dnsdbflex --regex "ibm" -l0 -j -O1000000 >> ibm-hits.jsonl**
$ **dnsdbflex --regex "ibm" -l0 -j -O2000000 >> ibm-hits.jsonl**
$ **dnsdbflex --regex "ibm" -l0 -j -O3000000 >> ibm-hits.jsonl**
$ **wc -l ibm-hits.jsonl**
4163008 ibm-hits.jsonl

We've now successfully retrieved over 4.16 million "ibm"-related results — BUT there may be still MORE "ibm" hits we can't access. That's the issue we run into if we "max out" the number of results that are retrievable — we don't know if there's one more result or 100 million more results out there over and above the ones we've been able to retrieve.

As a first step, let's begin by understanding the 4 million+ matches we *have* been able to successfully retrieve.

3. Exploring the "ibm" Domains We Have Discovered

At its most basic, some may try to get a sense of the sites we've found by simply "eyeballing" some of those four-million-plus domains.

Getting back to our results — each of the results will be on a line of their own, and consist of a domain name and an associated RRtype. Originally, Flexible Search results ALSO contained both "count" data for each record and "time first seen"/"time last seen data'' by default, which is why you'll see references in the dnsdbflex manual page to "terse" mode, even though that's now the default mode (and is, in fact, the *only* currently available Flexible Search mode).

If you're not a long-time Un*x user, one of the challenges with working at the command line can be learning Un*x command line commands that you may need to do your analyses. As part of this writeup, we're going to show you the Un*x commands we actually routinely use. Two common Un*x commands are the head and tail commands:

- The head command shows us ten lines from the start of the file.
- The tail will show us ten lines from the end of the file.

We'll use the Un*x grep command with the --color option to search for, and highlight, our target string in what gets output (one "real dot" replaced with [dot] in the following):

$ **head ibm-hits.jsonl | grep --color "ibm"**
{"rrname":"**ibm**dev1.e[dot]ac.","rrtype":"A"}
{"rrname":"**ibm**prod4.e[dot]ac.","rrtype":"A"}
{"rrname":"**ibm**c-cnrs.unistra.fr[dot]ac.","rrtype":"A"}
{"rrname":"qzdxw**ibm**a.gt[dot]ac.","rrtype":"A"}

{"rrname":"acyhvhlaibmq.gt[dot]ac.","rrtype":"A"}
{"rrname":"habibmonji.hv[dot]ac.","rrtype":"A"}
{"rrname":"fibms.in[dot]ac.","rrtype":"A"}
{"rrname":"ftibm.in[dot]ac.","rrtype":"A"}
{"rrname":"gibmo.in[dot]ac.","rrtype":"A"}
{"rrname":"gtibm.in[dot]ac.","rrtype":"A"}

$ **tail ibm-hits.jsonl | grep --color "ibm"**

{"rrname":"stgkibmprod4.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgnlibmprod.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgq-ibmprod.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgsibmprod4.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgt-ibmprod.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgtibmprod4.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgu-ibmprod.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgyibmprod4.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stgzibmprod4.wan.gs[dot]com.","rrtype":"TXT"}
{"rrname":"stpeibmprod4.wan.gs[dot]com.","rrtype":"TXT"}

As you can see in the above colorized output, each result does indeed have the requested string "ibm". So far, it doesn't look as if any of those domains are self-evident and undeniable "smoking gun" indicators that someone's leveraging IBM's registered marks for phishing or other nefarious purposes.

That said, we only have "peeked at" 20 results out of over four million. Let's try a more systematic review. Let's figure out:

a) What top level domains (TLDs) were those results from? Were they only from the "ac" to "com" TLDs, perhaps?
b) The first and last results shown above include "A" and "TXT" records — are those the *only* record types included?
c) How *long* are the RRnames we found? Are we finding primarily short names? If we could have gotten more results, would we have gotten longer domain names?
d) How many labels (or "dot separated parts") are present for a typical RRname we found? Two labels? Three? Four? More?

We'll explore those four analyses in the order listed.

4. Top Level Domains Found in Our 4.16 Million Results

To find the top level domains in our results, we'll extract the RRnames with jq (see <https://stedolan.github.io/jq/>) then pump those names through a pipeline of Un*x commands to summarize our results:

$ **jq -r '.rrname' < ibm-hits.jsonl | 1st-level-dom | sort | uniq -c | sort -nr > ibm-hits-tlds.txt**

Decoded:

jq	This is a "Swiss Army knife"-like tool for processing JSON Lines output
-r	This jq option asks for "raw" output (output without double quote marks)
'.rrname'	This is the named field we want
< filename	This passes in a file of records to process
|	Pipe the output from the previous command as input to the next command
1st-level-dom	Small script to extract just the effective top-level-domain (see Appendix I)
sort	Sort the lines to group the various TLDs in order
uniq-c	Count each unique TLD value we observe
sort-nr	Sort the counts and associated TLDS in reverse numeric order
> filename	Save the output to the specified filename

Our output from that script shows a broad range of domains, including many dot com domains (as one would expect, given that dot com is by far the most popular Internet TLD), but also numerous dot cn and dot ru domains, and even a bunch of dot dk domains:

$ **more ibm-hits-tlds.txt**
546683 com
409155 cn
408961 ru
282883 dk
211165 de
197146 io
161036 us
105962 com.cn
96983 nl
94270 co.uk
82511 com.br
74372 co
65718 biz
59243 me
55491 be
50653 tk
49840 es
48690 cc
46008 pl

43108 fr
34299 jp
33651 com.au
31993 ca
[etc]

Note: Some of the "TLDs" shown in the extract shown above are actually "effective TLDs." That is, those are "two part" domain suffixes listed in the Public Suffix List (see <https://publicsuffix.org/>) that "act as if" they are TLDs (e.g., approved parties can register domains under those "two label suffixes" even though they have two parts rather than just the normal single part).

Anyhow, we can clearly see that we have results from a wide variety of top level (and effective top level) domains. We're *not* just seeing results from a subset of TLDs, such as only TLDs from "ac" to "com."

5. The Resource Record Types We've Found

Let's now see what record types we've found. Our "eyeball" inspection revealed "A" (domain name to IPv4 address) records and "TXT" records, but what do we see if we look more systematically?

We'll use a process similar to the approach we used in the analysis shown in part 4, this time focusing on the .rrtype field:

$ **jq -r '.rrtype' < ibm-hits.jsonl | sort | uniq -c | sort -nr > ibm-hits-rrtypes.txt**
$ **more ibm-hits-rrtypes.txt**
2160651 A
1207998 CNAME
553586 TXT
70334 SOA
48672 NS
47096 AAAA
32354 NULL
21291 MX
16897 WKS
2931 HINFO
476 TYPE65
217 SRV
158 KEY
98 SPF
82 CAA
76 RP
41 ANY
18 PTR
10 LOC

6 TYPE65399
5 DHCID
4 APL
3 DNAME

That's not a particularly unusual distribution of RRtypes, with (2160651+1207998)/4163008*100=80.9% of all records consisting of "A" or "CNAME" records (more on the top RRtypes: <https://www.farsightsecurity.com/blog/txt-record/dnsrecords-20171201/>).

It might be short sighted (and may result in us overlooking interesting names seen only in some of the more obscure RRtypes), but if we *really* don't care about less-common record types, we might consider limiting our search to just "A" and "CNAME" records, thereby reducing our results by nearly 20% via that single exclusionary step.

6. Domain Length in Characters

We also wanted to get a sense of the size distribution of the names we saw. Are they all "reasonable" looking "short-length" names, perhaps? Or are some improbably long names? The commands to find out should now be looking fairly familiar, since this is just a variation of the approach we've previously demonstrated:

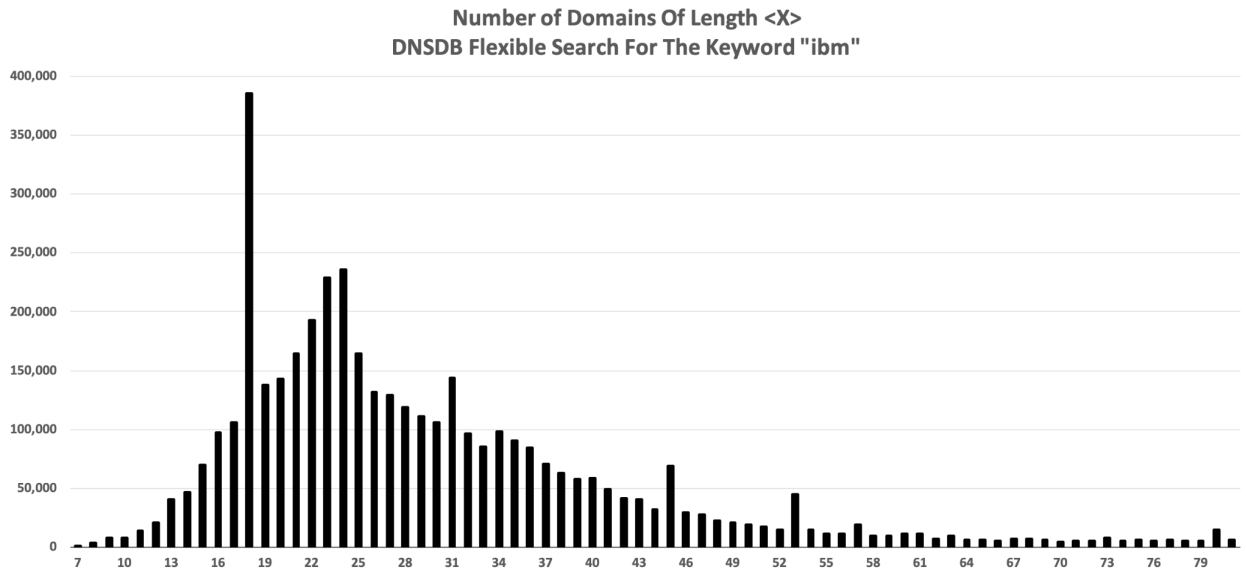
$ **jq -r '.rrname' < ibm-hits.jsonl | awk '{ print length }' | sort -n | uniq -c > ibm-hits-domain-lengths.txt**

$ **more ibm-hits-domain-lengths.txt**
560 7
3560 8
7519 9
7614 10
13706 11
20551 12
40113 13
46153 14
69708 15
96933 16
105617 17
384695 18
137970 19
142885 20
164587 21
192180 22
228809 23
235303 24
163889 25
131197 26

128947 27
118602 28
111187 29
105772 30
143519 31
96764 32
84960 33
98474 34
90031 35
83979 36
70713 37
63254 38
57382 39
58245 40
49343 41
41581 42
40275 43
31867 44
68739 45
29710 46
27263 47
22635 48
20499 49
19192 50
17219 51
15145 52
45096 53
14927 54
11160 55
11589 56
18620 57
9576 58
9630 59
11438 60
11156 61
7181 62
9375 63
6423 64
5861 65
5583 66
7141 67
7273 68
6155 69

4132 70
5006 71
5652 72
7546 73
4934 74
5801 75
4891 76
6026 77
5128 78
5695 79
14823 80
6344 81

You may notice that the above distribution is truncated at 81 characters and wonder why. The answer is that Flexible Search was designed/programmed to not index domain names over 81 characters in length (some junk domain names may get absurdly long). It's easiest to get a sense of that table of values by graphing the data with Excel (or another graphics package of your choice):



The spike (at length 18) is a particularly obvious anomaly. Let's check to see if that large spike is largely attributable to one (or perhaps just a couple) effective 2nd-level domains (we've replaced one "real dot" in each name with [dot]):

$ jq -r '.rrname' < ibm-hits.jsonl | awk 'length==18' | 2nd-level-dom | sort | uniq -c | sort -nr > ibm-top-length-18-hits.txt
$ more ibm-top-length-18-hits.txt
268960 tv2[dot]dk <== 268960/384695*100=69.9% of names with length=18 [highlight added manually]
3688 58[dot]com

1997 joybuy[dot]es
1819 mail[dot]ru
1742 ibmuyym[dot]cn
1044 gs[dot]com
948 tvingo[dot]ru
872 yandex[dot]ru
872 ibm[dot]cm
819 mibmofd[dot]cn
777 bsxlibm[dot]cn
597 esu[dot]cn

In this case, tv2[dot]dk domains look like they are the dominant cause of that spike, while also contributing heavily to the prominence of the dot dk TLD that we saw in section 4 of this article.

Looking at just those "tv2[dot]dk" domains, they look as if they may have "effectively random" (or at least "opaquely-encoded") hostname parts:

$ **jq -r '.rrname' < ibm-hits.jsonl | awk 'length==18' | fgrep ".tv2[dot]dk" | head**
001jrsuibm.tv2[dot]dk.
002ee64ibm.tv2[dot]dk.
004iylibmr.tv2[dot]dk.
004rfibmyb.tv2[dot]dk.
005bqibmpk.tv2[dot]dk.
007ibmvwmu.tv2[dot]dk.
008o4ibmwm.tv2[dot]dk.
008skibmwj.tv2[dot]dk.
00bkibmzgl.tv2[dot]dk.
00cibmhmbk.tv2[dot]dk.

$ **jq -r '.rrname' < ibm-hits.jsonl | awk 'length==18' | fgrep ".tv2[dot]dk" | tail**
zzy5r2iibm.tv2[dot]dk.
zzybtuibmw.tv2[dot]dk.
zzyhdcibmr.tv2[dot]dk.
zzyibm8fdm.tv2[dot]dk.
zzyibmmqrj.tv2[dot]dk.
zzysd10ibm.tv2[dot]dk.
zzytbntibm.tv2[dot]dk.
zzzeibmpkv.tv2[dot]dk.
zzzibmyyc5.tv2[dot]dk.
zzzpibmzc2.tv2[dot]dk.

Those aren't the only detectable anomalies. For instance, what about the names that are "quite long?" Most users wouldn't interactively enter a domain name that's 72 or more characters long, would they? My gosh, the potential

for typographical errors in that scenario, eh? Naturally, some domain names may be used programmatically (rather than just interactively by end users), but nonetheless, those names are still "strange" even for "back end"/"infrastructural" domain names.

Let's see if we've got many of those:

$ **jq -r '.rrname' < ibm-hits.jsonl | awk 'length>71' > ibm-over-71.txt**
$ **wc -l ibm-over-71.txt**
66840 ibm-over-71.txt

So that tells us that we have nearly 67,000 fully qualified domain names (FQDNs) that are pretty darn long. Let's see what some of those look like (we've replaced one "real dot" in each of the following names with [dot]):

$ **more ibm-over-71.txt**
ahr0cdovl3bsyw50z2vuzxjhlm9yzy9jtexvu1rsqvrjt05tx3rodw1ibmfpbhm.netflix[dot]ac.
ahr0cdovl3d3dy5uyxr1cmfsbwvkawnpbmfsagvyynmubmv0l3rodw1ibmfpbhm.netflix[dot]ac.
lelrhsbibmgy._tcp.default-first-site-name._sites.dc._msdcs.shuaacapital[dot]co.ae.
portal-ssl1898-19.affable-mongodb-51-backup-version-test.ibm-495.gap[dot]ae.
portal152-0.elasticsearch-conversation-premium-slot-pmp009.ibm-watson.gap[dot]ae.
bmix-dal-ys1-f08c024c-145f-4cea-8a44-92c02fced5ff.ibm-bmix-sandbox.gap[dot]ae.
bmix-dal-yp-c7e36b1e-cf5c-4afc-831e-8d84b4c0afc8.liorlu-il-ibm-com.gap[dot]ae.
bmix-dal-yp-79c2dc06-1848-4775-90f1-ad7a95ca91d2.ffbld01-uk-ibm-com.gap[dot]ae.
bmix-dal-yp-82741511-0868-4ec0-b8cc-e8cc957c1702.ffbld01-uk-ibm-com.gap[dot]ae.
bmix-dal-yp-a1a5d50f-7fee-4f71-b510-13acac3fe4b6.ffbld01-uk-ibm-com.gap[dot]ae.
bmix-dal-yp-c81d6a34-261b-4028-a709-7026975e0de7.zhangce-us-ibm-com.gap[dot]ae.
bmix-dal-yp-15445f48-3cf7-4fbe-9825-15845364a92a.bakribbs-us-ibm-com.gap[dot]ae.
bmix-dal-yp-14797224-352d-4789-aed7-63878099a776.bsrinivk-in-ibm-com.gap[dot]ae.
bmix-dal-yp-994c65f2-bd18-465b-8f7f-99249c8ff372.ffprod01-uk-ibm-com.gap[dot]ae.
bmix-lon-yp-1e65e845-adef-43b3-a85c-b85d603828b2.pskhadke-us-ibm-com.gap[dot]ae.
[etc]

Some of those look as if they may incorporate UUID strings, but many "long names" of this sort are often "wildcard" domains that include random gibberish — domains which will resolve anything/everything used for the "hostname" part of the FQDN.

Let's try making up some random hostname parts to test a few of the the domains shown above (one "real dot" replaced with [dot] in the following):

$ **dig aoifjoaifgjoafsjafs.netflix[dot]ac +short**
detour.netflix[dot]net.
detour.prod.netflix[dot]net.
34.218.19.240
44.226.113.145

18.236.7.30

$ **dig oajifoaifsjoafjadsis.gap[dot]ae +short**
162.13.201.232

Yep, both those 2nd-level domains look like wildcard domains to us. We can probably safely filter those domains, not because they're "known good" or "known bad," but simply because they're likely NOT particularly focused on "IBM-related" mischief.

7. Domain Label Count

Finally, we also wanted to check the number of "labels" or "domain name parts" per-name in our results (for instance, "ibm.example.com." would have three labels while "www.ibm.example.com." would have four labels). To count the number of labels in each name, we'll use a little trick. Specifically, because each name ends in a "formal trailing dot", a name with three labels will also (conveniently!) have three dots, a name with four labels will have four dots, etc.

Taking advantage of that relationship, we'll use the Un*x sed ("stream editor") command to keep JUST the dots from each result, and then pipe those dots through awk to print the number of dots seen per name, just as we previously used awk to print the length of the names in part 6 of this article:

$ **jq -r '.rrname' < ibm-hits.jsonl | sed 's/[^\.]//g' | awk '{ print length }' | sort -n | uniq -c > ibm-labels-per-name.txt**
$ **more ibm-labels-per-name.txt**
106472 2
1633699 3
1412700 4
585924 5
211200 6
88532 7
77362 8
29575 9
8127 10
4542 11
1920 12
1132 13
674 14
466 15
382 16
201 17
80 18
18 19
2 20

We'd expect most names to be in the two-to-six label range, and that's actually most of what we see in that table. But 7-20 labels? That seems pretty crazy! Let's extract those for closer scrutiny:

$ **jq -r '.rrname' < ibm-hits.jsonl | egrep "(\..*){7,20}" > ibm-lots-of-labels.txt**
$ **wc -l ibm-lots-of-labels.txt**
213013 ibm-lots-of-labels.txt

Many-label-domain names really ARE present in our results! Peeking at those two hundred-thousand plus records, we see the following (note that we've replaced one "real dot" in each name with [dot]):
$ **more ibm-lots-of-labels.txt**
[* * *]
www.587231.www.414466.jibm[dot]ac.cn.
www.evvtyc.www.414466.jibm[dot]ac.cn.
www.vwrtrg.www.414466.jibm[dot]ac.cn.
www.288842.www.vawrzq.jibm[dot]ac.cn.
www.793618.www.vawrzq.jibm[dot]ac.cn.
www.qpzdgo.www.vawrzq.jibm[dot]ac.cn.
www.tcbpvr.www.vawrzq.jibm[dot]ac.cn.
www.zrddvd.www.vawrzq.jibm[dot]ac.cn.
www.580013.www.zofgmj.jibm[dot]ac.cn.
www.668959.www.zofgmj.jibm[dot]ac.cn.
www.718796.www.zofgmj.jibm[dot]ac.cn.
www.760662.www.zofgmj.jibm[dot]ac.cn.
www.770164.www.zofgmj.jibm[dot]ac.cn.
www.972741.www.zofgmj.jibm[dot]ac.cn.
www.bjofwe.www.zofgmj.jibm[dot]ac.cn.
www.buybqj.www.zofgmj.jibm[dot]ac.cn.
[* * *]
48foccibmsjc1.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
5.focc-ibm-sjc-1.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
ibm-dev-images.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
ibmdevtst-isus.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
iib-amc-ibmtoc.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
sonyibmhdctx1nat.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
suawindowsibmitam.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
ubuntuiem2ibmitam.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
cache-novosibmts06.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
centos2iem3-ibmitam.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
redhatiem2ibmitam-buh.ftbsitessvr01-main.ks.dev.consent.wpsites01[dot]ft.com.
[etc]

So, what we're seeing from all of the above analyses is that we basically have two objectives:

- We need to try to create a more narrowly-targeted query (thereby reducing the initial quantity of results found)

- We need to do "noise filtering" on our Flexible Search results (thereby dumping results that only incidentally include our target string as random noise).

Once we've done those two things, we'll be ready to enhance the remaining results we've discovered using DNSDB Standard Search.

8. Being Selective Rather Than Just "Brute Forcing" Our Standard Search Followup Queries — Why Bother?

Some might ask, *"Why not skip all the data analysis and just look up ALL the hits we've found in DNSDB Flexible Search in DNSDB Standard Search?"*

We **could** actually have just looked up all four million plus results we found, but that would a) take a while and b) consume a lot of DNSDB queries, which may be important for most folks who have a limited daily query quota. To put those realities in perspective:

- DNSDB API Standard Search query throughput will vary depending on:
  - The time it takes to establish an encrypted and authenticated connection, and to then transmit your query
  - The exact queries you make (some queries will get processed faster than others simply because of the number and type of results available)
  - Whether you run queries sequentially or via up-to-ten parallel streams
  - Server and network load from other customers (rarely much of a factor)
  - Realized network throughput to return your results (which can be impacted by things like network latencies and bandwidth-delay products, network loss/retransmissions (if any), as well as other factors).

For back-of-the-napkin analysis purposes, let's assume you complete between one query every ten seconds and one query every 1/10th of a second (your throughput may be within or outside that range — we'd encourage you to do your own benchmarking if you're planning on doing a large number of queries). Doing the basic arithmetic:

- (60 seconds/minute)*(60 minutes/hour)*(24 hours/day) = 86,400 seconds/day

- At one query every ten seconds, that would imply an ability to do 8,640 queries/day, while at one query every tenth of a second, that would imply the ability to do 864,000 queries/day, assuming you're submitting those queries in series rather than in parallel.

- If we wanted to complete roughly 4 million queries in no more than 24 hours, that would imply realizing throughput of 4,000,000/86,400=46.3 queries/second.

- Throughput considerations aside, there's also the matter of DNSDB quota availability. If you're purchased a DNSDB API license allowing unlimited queries per day (over up to ten parallel connections), you don't need to worry about your quota, but if you only purchased 10,000 queries/day, those could get consumed pretty quickly (and it wouldn't really be practical to try looking up four million results in DNSDB Standard Search at that low daily rate).

Pragmatically, we need to narrow-in-on our query.

9. Narrowing-In-On" What We REALLY Want to Analyze

We can tailor our query more narrowing in a number of different dimensions, some of which were hinted-at in the exploratory data analysis done in sections 4-7 above:

- **Time Fencing:** Rather than going back and looking at results from the full history of DNSDB (e.g., from June 2010 forward), let's look at just results seen during some recent time period, such as the last 90 days,the last month, or the last week. Time fencing this way should substantially reduce the number of domains we find, and give us results that are far more operationally-germane than results that may be from a decade ago.

- **Only Select the RRtypes Known to Be of Interest:** Rather than accepting all non-DNSSEC RRtypes, let's do multiple RRtype-specific queries (such as one for just "A" records, and another one for just "CNAMES"). In this example, that won't necessarily result in a huge reduction in volume (because we already know from section 8 above that ~80% of our results are likely to be "A" records or "CNAME" records), but if we're not interested in some of the more esoteric RRtypes, we might as well omit them. [On the other hand, this may multiply the number of queries you need to make, always a consideration in quota-constrained environments.]

- **Exclude "Likely Good"/"Likely Safe"/"Likely Irrelevant" Effective 2nd-Level and Top-Level Domains:** Let's also see if we can't identify some 2nd-level domains that we probably don't need to worry about if we're a brand manager or company security analyst.

  For example, we're probably safe in assuming that domains under ibm.com (e.g., IBM's own primary domain) won't be something we need to worry about, ditto any dot gov or dot mil domains (although obviously even the most carefully-managed of TLDs may still potentially end up victimized). Similarly, some domains that are only seen as part or a reflective DNS amplification DDoS attack can also probably be safely ignored if our interests are solely phishing or brand abuse.

- **"Abnormal Length"/"High Label Count" Domains:** Let's junk/deprioritize these, too. They may be an interesting curiosity to look at eventually, but for our first pass analysis, we've got more important things to worry about first.

We'll call this initial set of filters our "screens." Let's now apply those screens to our study.

10. Screen #1: Time Fencing

As a first goal, we'd like to get to the point where we have less than 1,000,000 results returned, thereby ensuring that we'll have ALL the results that are available, albeit for a shorter interval.

Let's begin by just asking for domains seen sometime during the last 90 days:

$ **dnsdbflex --regex "ibm" -l0 -j -A90d > ibm-hits-last-90.jsonl**
$ **wc -l ibm-hits-last-90.jsonl**
1142816 ibm-hits-last-90.jsonl

1,142,816 results means that we still have a huge number of hits. We could ask for additional tranches of results, as previously shown, let's try a shorter period, like just the last 30 days, instead?

$ **dnsdbflex --regex "ibm" -l0 -j -A30d > ibm-hits-last-30.jsonl**
$ **wc -l ibm-hits-last-30.jsonl**
1221628 ibm-hits-last-30.jsonl

Nope, still too many results. How about just results from the last 7 days?

$ **dnsdbflex --regex "ibm" -l0 -j -A7d > ibm-hits-last-7.jsonl**
$ **wc -l ibm-hits-last-7.jsonl**
484331 ibm-hits-last-7.jsonl

NOW we're well under our target value. We might actually be able to expand our time fencing slightly (e.g., from seven days to eight or nine or ten days) while still staying under a million results, but a week represents a nice "round number" for the purposes of this example. We'll stick with that time fence.

11. Screen #2: Resource Record Types

Now we'll refine our queries to just ask for specific record types of interest. In our case, let's assume that we only care about "A" records and "CNAME" records (you may have different record type interests, and if so, that's fine. The process is similar regardless of whether you're interested in "AAAA" records, "MX" records, "TXT" records, etc.).

When using flexible search, we'll need to make **separate requests for each record type we want to specifically request:**

$ **dnsdbflex --regex "ibm" -l0 -j -A7d -t a > ibm-hits-last-7-a-only.jsonl**
$ **wc -l ibm-hits-last-7-a-only.jsonl**
281967 ibm-hits-last-7-a-only.jsonl

$ **dnsdbflex --regex "ibm" -l0 -j -A7d -t cname > ibm-hits-last-7-cname-only.jsonl**

$ **wc -l ibm-hits-last-7-cname-only.jsonl**
121510 ibm-hits-last-7-cname-only.jsonl

If we concatenate those two output files and keep only unique RRnames, we end up with 403,287 hits, a reduction of 81,044 records over our original 7 days worth of results:

$ **cat ibm-hits-last-7-a-only.jsonl ibm-hits-last-7-cname-only.jsonl | jq -r '.rrname' | sort -u > ibm-hits-last-7-a-and-cname-only.jsonl**
$ **wc -l ibm-hits-last-7-a-and-cname-only.jsonl**
403287 ibm-hits-last-7-a-and-cname-only.jsonl

12. Screen #3: Exclude Domains Owned by IBM Itself, and Any (Presumed-To-Be-OK) Governmental/Educational Effective TLDs

We're going to assume that IBM itself watches domains under their own 2nd-level domains. So we're not going to worry about any domains in .ibm.com (or in .ibmcloud.com or .ibmcollabcloud.com). We'll match those names using an extended regular expression and the Un*x egrep ("extended grep") command:

$ **egrep "(\.ibm\.com\.$|\.ibmcloud\.com\.$|\.ibmcollabcloud\.com\.$)" ibm-hits-last-7-a-and-cname-only.jsonl | wc -l**
192321 *<== we're going to ignore all of these...*

Decoding that regular expression:

- We've got three patterns of interest, and we want to match any records containing ANY of those three
- We group those alternatives within parentheses, separated by logical "OR" symbols ("vertical bar" symbols)
- "Raw dots" will normally match any one character, so we'll backslash "real" (literal) dots where they appear in the patterns
- We end each of our three alternative patterns with a dollar sign, or "right hand anchor", so we don't end up inadvertently matching the specified patterns if it appears in the "middle" of a domain
- We embed the whole thing in double quote marks so the Un*x shell doesn't interfere with interpretation of our pattern

That command looks for affirmative (positive) matches. We can also use the dash vee to invert the sense of the match, so that only lines that would NOT match get selected as output:

$ **egrep -v "(\.ibm\.com\.$|\.ibmcloud\.com\.$|\.ibmcollabcloud\.com\.$)" ibm-hits-last-7-a-and-cname-only.jsonl > ibm-hits-last-7-a-and-cname-only-wo-ibm-internal.jsonl**

$ **wc -l ibm-hits-last-7-a-and-cname-only-wo-ibm-internal.jsonl**

210966 ibm-hits-last-7-a-and-cname-only-wo-ibm-internal.jsonl *<== this is what's left*

Now, let's extract a list of any effective TLDs that look academic/educational (as a former academic, we're fairly comfortable assuming that academic network operators/system administrators are paying attention to their kit, and not allowing anything too bizarre to happen on their networks/systems).

In the United States, academic/educational names will often be either in the dot edu TLD, or be a k12.<state>.us domain, but when we go abroad, academic domains are often tagged with ac instead. We'll find potentially matching domains of that sort with:

$ **grep "ac\." ibm-hits-tlds.txt > exclude-ac.txt**
$ **grep "edu" ibm-hits-tlds.txt > exclude-edu.txt**
$ **grep "k12" ibm-hits-tlds.txt > exclude-k12.txt**

We are similarly going to trust government and military domains by default. In the United States, governmental or military domains implies dot gov or dot mil, but overseas sites may tag those domains with go, gob, gouv, or police instead (among other things):

$ **grep "go\." ibm-hits-tlds.txt > exclude-go.txt**
$ **grep "gob" ibm-hits-tlds.txt > exclude-gob.txt**
$ **grep "gov" ibm-hits-tlds.txt > exclude-gov.txt**
$ **grep "gouv" ibm-hits-tlds.txt > exclude-gouv.txt**
$ **grep "mil" ibm-hits-tlds.txt > exclude-mil.txt**
$ **grep "police" ibm-hits-tlds.txt > exclude-police.txt**

We manually reviewed all the patterns that resulted from those searches, manually deleted a few spurious matches, and assembled the results into the following (admittedly somewhat daunting appearing!) consolidated regular expression:

(\.ac\.cn\.$|\.ac\.uk\.$|\.ac\.jp\.$|\.ac\.id\.$|\.ac\.kr\.$|\.ac\.il\.$|\.ac\.in\.$|\.ac\.ae\.$|\.ac\.za\.$|\.ac\.th\.$|\.ac\.at\.$|\.ac\.nz\.$|\.ac\.be\.$|\.ac\.ir\.$|\.ac\.tz\.$|\.ac\.ke\.$|\.ac\.rs\.$|\.ac\.ug\.$|\.ac\.pa\.$|\.ac\.lk\.$|\.ac\.me\.$|\.ac\.rw\.$|\.ac\.ru\.$|\.ac\.mz\.$|\.ac\.bd\.$|\.ac\.mu\.$|\.ac\.cy\.$|\.edu\.cn\.$|\.edu\.tw\.$|\.edu\.ph\.$|\.edu\.in\.$|\.edu\.tr\.$|\.edu\.vn\.$|\.edu\.my\.$|\.edu\.au\.$|\.edu\.ua\.$|\.edu\.pl\.$|\.edu\.br\.$|\.edu\.ws\.$|\.edu\.sa\.$|\.edu\.bd\.$|\.edu\.pk\.$|\.edu\.qa\.$|\.edu\.np\.$|\.edu\.ec\.$|\.edu\.mx\.$|\.edu\.ar\.$|\.edu\.ge\.$|\.edu\.hk\.$|\.edu\.lk\.$|\.edu\.sg\.$|\.edu\.do\.$|\.edu\.co\.$|\.edu\.ng\.$|\.edu\.iq\.$|\.edu\.mk\.$|\.edu\.bo\.$|\.edu\.pe\.$|\.edu\.gh\.$|\.edu\.eg\.$|\.edu\.zm\.$|\.edu\.ve\.$|\.edu\.sy\.$|\.edu\.ms\.$|\.edu\.kh\.$|\.edu\.jo\.$|\.edu\.cu\.$|\.edu\.az\.$|\.edu\.af\.$|\.edu\.om\.$|\.edu\.mn\.$|\.edu\.lb\.$|\.edu\.it\.$|\.edu\.al\.$|\.edu\.sv\.$|\.edu\.ru\.$|\.edu\.rs\.$|\.edu\.pt\.$|\.edu\.lv\.$|\.edu\.kz\.$|\.edu\.kw\.$|\.edu\.ee\.$|\.edu\.bn\.$|\.edu\.ba\.$|\.go\.id\.$|\.go\.pw\.$|\.go\.kr\.$|\.go\.jp\.$|\.go\.th\.$|\.go\.ke\.$|\.go\.ug\.$|\.go\.cr\.$|\.gob\.bo\.$|\.gob\.mx\.$|\.gob\.pa\.$|\.gob\.ve\.$|\.gob\.ar\.$|\.gob\.ni\.$|\.gob\.hn\.$|\.gob\.ec\.$|\.gob\.cl\.$|\.gob\.gt\.$|\.gob\.es\.$|\.gouv\.fr\.$|\.gov\.my\.$|\.wa\.gov\.au\.$|\.gov\.ph\.$|\.gov\.au\.$|\.gov\.cn\.$|\.gov\.ae\.$|\.gov\.uk\.$|\.govt\.nz\.$|\.gov\.ru\.$|\.gov\.vc\.$|\.gov\.tr\.$|\.gov\.pl\.$|\.df\.gov\.br\.$|\.gov\.in\.$|\.gov\.pk\.$|\.gov\.ua\.$|\.gov\.eg\.$|

.gov\.la\.$|\.gov\.it\.$|\.gov\.bd\.$|\.gov\.tw\.$|\.gov\.vn\.$|\.vic\.gov\.au\.$|\.sp\.gov\.br\.$|\.gov\.sa\.$|\.gov\.kw\.$|\.gov\.ie\.$|\.gov\.ng\.$|\.gov\.sg\.$|\.gov\.co\.$|\.gov\.jo\.$|\.gov\.ir\.$|\.gov\.kh\.$|\.gov\.br\.$|\.sr\.gov\.pl\.$|\.gov\.za\.$|\.tas\.gov\.au\.$|\.sa\.gov\.pl\.$|\.homeoffice\.gov\.uk\.$|\.gov\.kg\.$|\.gov\.ar\.$|\.um\.gov\.pl\.$|\.so\.gov\.pl\.$|\.gov\.kz\.$|\.gov\.il\.$|\.gov\.bs\.$|\.gov\.bn\.$|\.gov\.af\.$|\.ap\.gov\.pl\.$|\.wiw\.gov\.pl\.$|\.po\.gov\.pl\.$|\.gov\.sy\.$|\.gov\.rw\.$|\.gov\.om\.$|\.gov\.mu\.$|\.gov\.mn\.$|\.gov\.ly\.$|\.gov\.ec\.$|\.witd\.gov\.pl\.$|\.psse\.gov\.pl\.$|\.piw\.gov\.pl\.$|\.gov\.sd\.$|\.gov\.rs\.$|\.gov\.mz\.$|\.gov\.ma\.$|\.gov\.hk\.$|\.gov\.al\.$|\.to\.gov\.br\.$|\.sc\.gov\.br\.$|\.rn\.gov\.br\.$|\.rj\.gov\.br\.$|\.pr\.gov\.br\.$|\.mg\.gov\.br\.$|\.kppsp\.gov\.pl\.$|\.kmpsp\.gov\.pl\.$|\.gov\.ws\.$|\.gov\.tt\.$|\.gov\.pg\.$|\.gov\.lb\.$|\.gov\.iq\.$|\.gov\.gh\.$|\.gov\.gd\.$|\.gov\.dz\.$|\.k12\.ok\.us\.$|\.k12\.az\.us\.$|\.k12\.ca\.us\.$|\.k12\.ct\.us\.$|\.k12\.me\.us\.$|\.k12\.ga\.us\.$|\.k12\.or\.us\.$|\.k12\.tr\.$|\.k12\.nc\.us\.$|\.k12\.ma\.us\.$|\.k12\.in\.us\.$|\.k12\.id\.us\.$|\.k12\.nj\.us\.$|\.k12\.mi\.us\.$|\.k12\.wi\.us\.$|\.k12\.pa\.us\.$|\.k12\.ar\.us\.$|\.k12\.mn\.us\.$|\.k12\.il\.us\.$|\.k12\.va\.us\.$|\.k12\.tx\.us\.$|\.k12\.sc\.us\.$|\.k12\.ia\.us\.$|\.k12\.co\.us\.$|\.k12\.oh\.us\.$|\.k12\.ny\.us\.$|\.k12\.nv\.us\.$|\.k12\.ms\.us\.$|\.k12\.mo\.us\.$|\.k12\.vi\.$|\.k12\.nm\.us\.$|\.k12\.mt\.us\.$|\.k12\.la\.us\.$|\.k12\.ks\.us\.$|\.k12\.fl\.us\.$|\.k12\.al\.us\.$|\.mil\.ph\.$|\.mil\.in\.$|\.mil\.id\.$|\.mil\.do\.$|\.mil\.ve\.$|\.mil\.sy\.$|\.mil\.pl\.$|\.mil\.ni\.$|\.mil\.ae\.$|\.police\.uk\.$)

While that expression looks long (and IS long), it only uses the regular expression functions we've already discussed. We could rewrite that expression in a more compact form, but since this article isn't focused on writing "optimal" or "compact '' regular expressions, we're going to just leave that expression in straightforward (if verbose!) form).

We can then use that regular expression to exclude governmental and educational domains:

$ **egrep -v -f all-exclusions.txt < ibm-hits-last-7-a-and-cname-only-wo-ibm-internal.jsonl > ibm-hits-last-7-a-and-cname-only-wo-ibm-internal-excluding-safe-tlds.txt**

$ **wc -l ibm-hits-last-7-a-and-cname-only-wo-ibm-internal-excluding-safe-tlds.txt**
208333 ibm-hits-last-7-a-and-cname-only-wo-ibm-internal-excluding-safe-tlds.txt

Admittedly, that was a fair bit of work to exclude only 210,966-208,333=2,633 FQDNs, BUT for those who may have only 10,000 queries per day, "every little bit" can be important (and we can easily re-use this pattern to help clean up other runs, too, although we'd likely need to double check to make sure we've got all the relevant patterns).

13. Screen #4: Excluding Too-Long Domains

We're now going to arbitrarily exclude all fully qualified domain names longer than 40 characters and all domains with six or more labels. These may seem like fairly "coarse" cuts to make, but in most cases these cuts are very effective when it comes to dumping noise while avoiding domains of substantive interest:

$ **awk 'length <= 40' < ibm-hits-last-7-a-and-cname-only-wo-ibm-internal-excluding-safe-tlds.txt > ibm-40-or-less.txt**
$ **wc -l ibm-40-or-less.txt**

152875 ibm-40-or-less.txt

$ **egrep -v "(\..*){6,}" < ibm-40-or-less.txt > ibm-40-or-less-with-no-more-than-five-labels.txt**
$ **wc -l ibm-40-or-less-with-no-more-than-five-labels.txt**
93544 ibm-40-or-less-with-no-more-than-five-labels.txt

14. So What's Left?

So, we've now gone from over 4.1 million domains down to just 93,544 domains. What's left? Well, there may still be more random-looking domain noise that we can safely dump.

In particular, note that some of the 93,544 domains consist of huge numbers of "variant versions" of individual effective 2nd-level domains. We can find those by saying:

$ **2nd-level-dom < ibm-40-or-less-with-no-more-than-five-labels.txt | sort | uniq -c | sort -nr > top-2ld-ibm-40-or-less.txt**
$ **wc -l top-2ld-ibm-40-or-less.txt**
23198 top-2ld-ibm-40-or-less.txt <== think of this as 23,198 "rows," one for each effective 2nd-level domain, with each row having a variant domain name count

The rows in that file look like (we've replaced one "real dot" with [dot] in each of these entries):

$ **more top-2ld-ibm-40-or-less.txt**
13430 kohlz[dot]com <== so this means that there were 13,430 unique FQDNs that were all just variants of kohlz[dot]com
5307 kohld[dot]com
5168 umantis[dot]com
2817 mybluemix[dot]net
1334 myonlinedata[dot]net
1332 stitchfix[dot]com
847 medallia[dot]com
725 coderpad[dot]io
682 ibmmobiledemo[dot]com
655 digitalframeflow[dot]com
542 oneclickfeed[dot]com
411 tradetalk[dot]us
390 elastic-cloud[dot]com
328 frontegg[dot]com
321 ibmpcug[dot]co.uk
264 vibmro[dot]com
261 ibmqd[dot]com
260 teamviewer[dot]com
252 lightning[dot]com

239 consumerdirectx[dot]com
235 roblox.com[dot]ru
216 datadoghq[dot]com
211 adidas[dot]com
[etc]

We're not going to show you all the 13,430 different kohlz.com domains that were "rolled up" into the one kohlz.com summary count above, but a **few** of those "kohlz.com" variants (this time in label-reversed and sorted format) look like:

$ **grep "kohlz.com" ibm-40-or-less-with-no-more-than-five-labels.txt | reverse-domain-name | sort**
com.kohlz.0-avatarcom.www.guardapibmnet
com.kohlz.05.com.www-services-ibmnet
com.kohlz.05.guardapibmnet-com
com.kohlz.05.guardapibmnet-com.www
com.kohlz.0earwww0com.www.guardapibmnet
com.kohlz.10-com.www.services-ibmnet
com.kohlz.14-miibmnet
com.kohlz.14-miibmnet.com
com.kohlz.14-miibmnet.com.www
com.kohlz.14.com-miibmnet
com.kohlz.14.com-miibmnet.www
com.kohlz.14.com.services-ibmnet
[...]
com.kohlz.com-comunidades.www.miibmnet
com.kohlz.com-connector.www-miibmnet
com.kohlz.com-core-com.www.miibmnet
com.kohlz.com-core.guardapibmnet-www
com.kohlz.com-core.www.bobjbibmnet
com.kohlz.com-core.www.com-cgibmnet
com.kohlz.com-core.www.comhrosapibmnet
com.kohlz.com-core.www.irishindibmnet
com.kohlz.com-core.www.webappusinibmnet
com.kohlz.com-corp.ibmnet-www
com.kohlz.com-corp.ibmnet-www.bmnet
com.kohlz.com-corp.www.appsusinibmnet
com.kohlz.com-corp.www.bmnet-cribmnet
[...]
com.kohlz.comepcapibmnettrack.www.bmnet
com.kohlz.comepcapibmnetybinst0
com.kohlz.comepcapibmnetybinst0.www
com.kohlz.comesom2.www.services-ibmnet
com.kohlz.comest.www-services-ibmnet

com.kohlz.comest.www.avatarapibmnet
com.kohlz.comest.www.bmnet-epcapibmnet
com.kohlz.comest.www.comdafitibmnet
com.kohlz.cometherpad.www.bmnetmiibmnet
com.kohlz.cometherpad.www.miibmnetbmnet
com.kohlz.cometherpad.wwwmiibmnet
[...]
com.kohlz.services-ibmnet-commlt5
com.kohlz.services-ibmnet-commlt5.www
com.kohlz.services-ibmnet-comnl
com.kohlz.services-ibmnet-comnl.www
com.kohlz.services-ibmnet-comoid
com.kohlz.services-ibmnet-comoid.www
com.kohlz.services-ibmnet-comsfmc
com.kohlz.services-ibmnet-comsfmc.www
com.kohlz.services-ibmnet-comteam
com.kohlz.services-ibmnet-comteam.www
com.kohlz.services-ibmnet-comus
[...]
com.kohlz.ybinst5.guardapibmnet.com
com.kohlz.ybinst5services-ibmnet
com.kohlz.ybinst5services-ibmnet.com
com.kohlz.ybinst6www-com.www.miibmnet
com.kohlz.ybinst9.services-ibmnet-com
com.kohlz.yourjourney-com.www.miibmnet

Why do those names exist? Are those names really pointing at substantive systems such as servers, workstations, networked printers, and other peripherals? No. This is another example of an effective 2nd-level domain that's a wildcard. ANYTHING we try to resolve that based on that domain WILL resolve, e.g. (note that in the following we've replaced one "real dot" with [dot]):

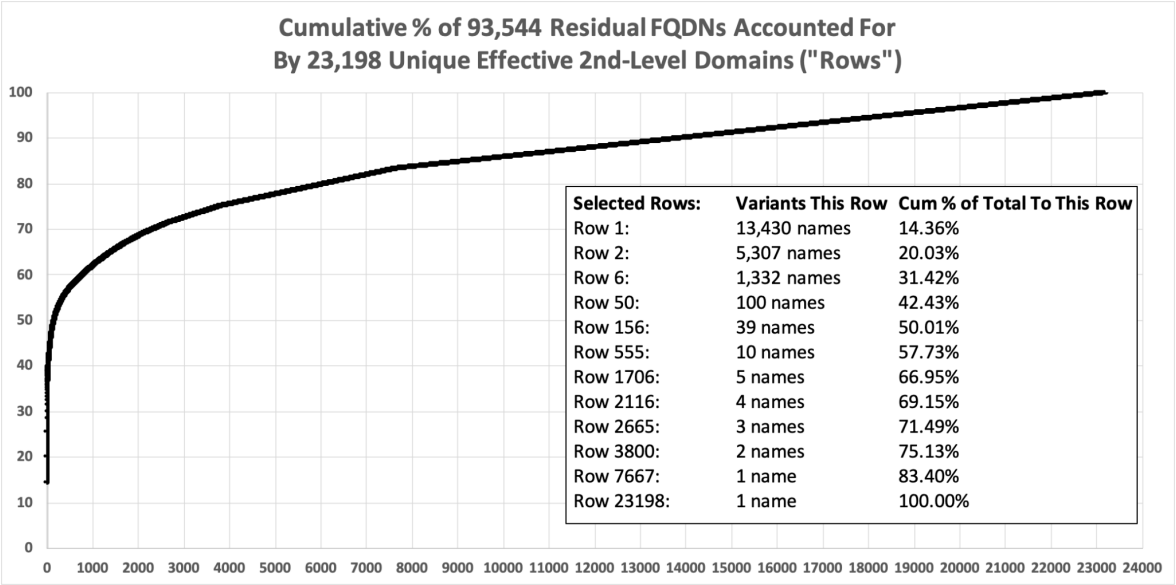
$ **dig oafjoaisfjoaisf.kohlz[dot]com +short**
103.224.182.252

$ **dig blahblahblahblah.kohlz[dot]com +short**
103.224.182.252

$ **dig lots-of-snow-this-month.kohlz[dot]com +short**
103.224.182.252

It is highly unlikely that any of these results represents a targeted attack on the ibm mark (e.g., as part of a phishing attempt, etc.).

Most times, wildcards are used as part of a DDoS attack or for tracking-related purposes, both of which are out of scope for this article. So, we looked at one potential wildcard-like effective-2nd-level domain, kohlz[dot]com — but what about the rest of the rows in our summary? We COULD test all 23,198 effective-level-domains for wildcarding, but we may not need to. JUST THE TOP 156 effective-2nd-level domains (out of a total of 23,198 effective-level-domains) collectively account for HALF of all our residual FQDNs, see the following cumulative distribution curve:



If we proceed to wildcard-test the top 156 effective-2nd-level domains, we find that 115 of those (73.7%) ARE wildcards (and thus represent names that we can filter as likely noise/incidental junk).

Some may wonder how we tested those names for "wildcard-ness." There are many ways to potentially do this, but one simplistic approach is to try resolving a synthetically-generated "random" hostname based on the 2nd-level domain of interest. If the domain is fully wildcarded, any such name will resolve. We'll use a primitive little Python3 script to generate the random hostname part:

```
#!/usr/local/bin/python3
import string
import random
def id_generator(size=16, chars=string.ascii_lowercase + string.digits):
        return ''.join(random.choice(chars) for _ in range(size))
print(id_generator())
```

We could then either write another little script to loop through a file of names to test, or simply cut and paste-together what we need. Since this is a "one-off" and involves only a couple pages of names, we'll use the latter approach (we've replaced one "real dot" in each name with [dot]):

```
host `./python_random`[dot]kohlz.com
```

host `./python_random`[dot]kohld.com
host `./python_random`[dot]umantis.com
host `./python_random`[dot]mybluemix.net
host `./python_random`[dot]myonlinedata.net
host `./python_random`[dot]stitchfix.com
host `./python_random`[dot]medallia.com
[etc]
This is an admittedly primitive approach, but sufficient for testing our 156 2nd-level names.

Some may wonder, *"WHY do you consider this to be a 'primitive' approach?"* We say that for a variety of reasons, including (but not limited to):

- The "host" command is meant as an interactive command, it isn't an API call. Scripting it is a kludge.
- The "host" command can't be tailored when it comes to things like timeouts — this means that if one of the authoritative name servers handling these domains wanted to, it can block subsequent queries for longer than we might prefer.
- We're composing the hostname using our little Python3 script, invoked line-by-line.
- We generally don't like to execute commands inline with backticks (even though in this case we have total control over what's passed for execution, and we can affirmatively ensure those commands aren't potentially dangerous).
- Our output is a plain text file, rather than something that's easier to parse, such as JSON Lines format.

Nonetheless, this approach shown above WILL work. Let's focus on our output. It looks like the following (we've replaced one "real dot" with [dot] in each of the following names):

psr4xdnzc1anhxtt.kohlz[dot]com has address 103.224.182.252
psr4xdnzc1anhxtt.kohlz[dot]com mail is handled by 10 park-mx.above.com.
lqdsc5rfooxhdich.kohld[dot]com has address 103.224.182.252
lqdsc5rfooxhdich.kohld[dot]com mail is handled by 10 park-mx.above.com.
rueov6o2mcyd2u7a.umantis[dot]com has address 185.238.12.10
y5nfbai0e4lq1ehg.mybluemix[dot]net has address 169.46.89.149
y5nfbai0e4lq1ehg.mybluemix[dot]net has address 169.47.124.22
y5nfbai0e4lq1ehg.mybluemix[dot]net has address 169.62.254.79
Host vsf70dqao173dvy8.myonlinedata[dot]net not found: 3(NXDOMAIN)
Host gt5ne2cjzjb82o30.stitchfix[dot]com not found: 3(NXDOMAIN)
Host 0m2wo8ulg74cv4pc.medallia[dot]com not found: 3(NXDOMAIN)
t29g90180c1if44y.coderpad[dot]io is an alias for wildcard.coderpad.io.herokudns.com.
wildcard.coderpad.io.herokudns[dot]com has address 54.237.159.171

wildcard.coderpad.io.herokudns[dot]com has address 3.226.182.14
wildcard.coderpad.io.herokudns[dot]com has address 52.21.227.162
wildcard.coderpad.io.herokudns[dot]com has address 23.22.5.68
h54ycolhqpnw3hop.ibmmobiledemo[dot]com has address 165.160.15.20
h54ycolhqpnw3hop.ibmmobiledemo[dot]com has address 165.160.13.20
r4yha50hkua2z2wl.digitalframeflow[dot]com has address 54.192.73.18
r4yha50hkua2z2wl.digitalframeflow[dot]com has address 54.192.73.104
r4yha50hkua2z2wl.digitalframeflow[dot]com has address 54.192.73.60
r4yha50hkua2z2wl.digitalframeflow[dot]com has address 54.192.73.127
[etc]

Those results are of four basic types:

- Evidence **REFUTING** Wildcardness: Some of the results may show a name didn't resolve ("NXDOMAIN"), supporting the conclusion that that particular 2nd-level domain is NOT a wildcard. We had 39 of those in our 156 domains test set.

- Evidence **SUPPORTING** Wildcardness: The other possibility is that we ARE able to successfully resolve our random 16 character long test name — those give us evidence supporting the conclusion that that domain is a wildcard (there's a vanishingly-small chance that a non-wildcard name will successfully resolve a randomly selected 16 character long hostname of our choice). 117 of the 156 domains were of that sort.

- Complications Associated with CNAMEs: The above two cases account for all 156 domain names, but NOT all of our output. Specifically, sometimes a wildcard exists, but is implemented via a CNAME (this is identifiable in the output when "is an alias for" gets mentioned). For example, this is the case for coderpad.io When a CNAME's involved, resolution of the name via the host command may result in "new domains" appearing in the left most column of our output (for example, wildcard.coderpad.io.herokudns.com). Those "new domains" represent names we need to "cull" from our output to avoid skewing things like our computation of the percentage of domains that are/aren't wildcarded.

- Results For Record Types We're Not Interested In: Finally, the host command may report on things we don't really care about, like MX (mail handler) records for the domain we tested. Those are another example of unneeded lines we'll want to remove.

We're now going to remove all FQDNs referring to any of the 117 determined-to-be-wildcard 2nd-level domains that appear in the 93,544 FQDNS in our ibm-40-or-less-with-no-more-than-five-labels.txt file, plus a few "bonus" exclusions we noticed while manually eyeballing what's left. We're going to do that by putting the patterns to be excluded in a file with a format that looks like the following (note that we've replaced one "real dot" with [dot] in each of the following):

$ **cat 117-to-filter.txt**

```
\.acronis\[dot]sport\.$
\.adidas\[dot]agency\.$
\.authenticatetrustpilot\[dot]com\.$
\.bingoservices\[dot]io\.$
\.boxde\[dot]com\.$
\.cingularextras\[dot]com\.$
\.cingularrefill\[dot]com\.$
\.clickflowzzz\[dot]com\.$
\.cloudbackupagent\[dot]com\.$
\.coderpad\[dot]io\.$
\.consumer-authtrustpilot\[dot]com\.$
\.consumerdirectx\[dot]com\.$
[...]
\.myonlinedata\[dot]net\.$
\.fly\.iberiaexpress\[dot]com\.$
\.splicex\.ibm-garage\[dot]com\.$
\.whs\.adidas\[dot]com\.$
\.daylight\.stitchfix\[dot]com\.$
\.shoaibmalik\[dot]ca\.$
\.go\.jetswap\[dot]com\.$
\.fc\.lazada\[dot]com\.$
\.container\.lightning\[dot]com\.$
\.nfi\[dot]com\.$
\.teamviewer\[dot]com\.$
\.hotel-hamburg-zentrum\[dot]de\.$
\.attalascom\[dot]net\.$
\.ibm\[dot]net\.$
^\_
```

Be sure there are NO BLANK LINES in that pattern file (blank lines will overmatch and result in EVERYTHING being filtered). We'll then apply the above filter rules by saying:

```
$ egrep -v -f 117-to-filter.txt < ibm-40-or-less-with-no-more-than-five-labels.txt >
whats-left-after-117-filtering.txt
$ wc -l whats-left-after-117-filtering.txt
49945 whats-left-after-117-filtering.txt
```

We have now gone from over 4,000,000 results to under 50,000.

Phase II: DNSBD Standard Search Phase

15. Running A Few of Our Remaining Domains Through DNSDB Standard Search

We're now ready to run a few of our remaining domains through DNSDB Standard Search. We'll use dnsdbq to do that (see <https://github.com/dnsdb/dnsdbq>). Running some FQDNs through DNSDB Standard Search will give us:

- The number of times each FQDN was seen (the "count")
- The time each FQDN was first seen (and last) seen, up to the time we did those runs
- The Rdata ("right hand side") associated with each of those FQDNs
- We can also ask for the IPs to be automatically mapped to the autonomous system that originates them. That will make it easier for us to do things like find (and exclude) domains announced by IBM's own network infrastructure.

We can do that several different ways.

We've previously described one approach to running batch queries, but 9/nother approach is to simply open the list of remaining domains in a text editor (such as vim or emacs) and add suitable text to the front and end of each line. That will create a little "script" you can use to sequentially run a batch of domains through DNSDB Standard Search. For example, let's pretend we had a file of just five domains to explore in DNSDB Standard Search (shown here with [dot] replacing the real dot just before the TLD):

5r3be6.s**ibm**az[dot]ru.
5t**ibm**.healthinshape[dot]com.
5u**ibm**.misroti[dot]com.
5yf39.**ibm**-oa[dot]cn.
5y**ibm**5.ocgreenrealty[dot]com.

We could use a text editor to prepend:

dnsdbq -r

to the beginning of each of those lines. "dnsdbq -r" says, "invoke the dnsdbq DNSDB Standard Search client" and "search RRnames in that database" (aka the "left hand side" of DNSDB records).

For example, you can add that text by saying (in the popular Un*x vim editor):

:1,$s/^/dnsdbq -r /

Your file should then look like the following (although you'd have "real dots" where we're showing [dot])

dnsdbq -r 5r3be6.sibmaz[dot]ru.
dnsdbq -r 5tibm.healthinshape[dot]com.
dnsdbq -r 5uibm.misroti[dot]com.
dnsdbq -r 5yf39.ibm-oa[dot]cn.
dnsdbq -r 5yibm5.ocgreenrealty[dot]com.

Now let's tack on some options to the end of each of those lines. We want to use:

-l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl

Translated, those options mean:

-l0	Request maximum results (e.g., up to a million results/query from dnsdbq)
-A7d	Just provide results seen sometime in the last 7 days
-j	Provide output in JSON Lines format
-a	ASN annotate the output with encompassing netblock and originating ASN
-T datefix,reverse,chomp	Give human-readable datetime formats, reverse RRnames by label, and remove the trailing dot (if one's present)
-t A	Only provide "A" records by way of response
>> results.jsonl	Append the results to the specified file

We can add that info in the vim editor by saying:

:1,$s/$/ -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl/

Your file should then look like (although you'd have "real dots" where we're showing [dot]):

dnsdbq -r 5r3be6.sibmaz[dot]ru. -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl
dnsdbq -r 5tibm.healthinshape[dot]com. -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl
dnsdbq -r 5uibm.misroti[dot]com. -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl
dnsdbq -r 5yf39.ibm-oa[dot]cn. -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl
dnsdbq -r 5yibm5.ocgreenrealty[dot]com. -l0 -A7d -j -a -T datefix,reverse,chomp -t A >> results.jsonl

You can then run that little file as a "script." For example, if the file is called test-run.bash, you'd say:

$ **chmod a+rx test-run.bash**
$ **bash test-run.bash**
Query status: NOERROR (no results found for query.)
Query status: NOERROR (no results found for query.)
Query status: NOERROR (no results found for query.)
Query status: NOERROR (no results found for query.)

In this case, note that an interesting thing happened: we ran five queries, but four of them found no results (just returning a warning message instead). This may be because we asked to only see results from the "last seven days," and there was a bit of a lag from finding the initial Flexible Search results and the time we began to shove those results through dnsdbq — those names may represent specific FQDN that were only seen once (or only rarely/infrequently).

The one result we DID receive looked like the following (note that the RRname gets displayed in "label-reversed" format since we used the -T reversed option):

```
$ jq < results.jsonl
{
 "count": 266,
 "time_first": "2021-03-06 18:11:43",
 "time_last": "2022-01-04 11:32:52",
 "rrname": "ru.sibmaz.5r3be6",
 "rrtype": "A",
 "bailiwick": "sibmaz[dot]ru.",
 "rdata": [
        "92.119.112.114"
 ],
 "dnsdbq_rdata": {
        "92.119.112.114": {
        "asinfo": {
        "as": [
        204601
        ],
        "cidr": "92.119.112.0/24"
        }
        }
 }
}
```

Looking at that result, we can pick out the various bits we'd hoped to find:

- The number of times this exact FQDN/RRtype/Bailiwick/Rdata combination was seen: 266
- The first seen and last seen datetimes: 2021-03-06 18:11:43 (UTC) to 2022-01-04 11:32:52 (UTC)
- The Rdata ("right hand side") associated with each of those FQDNs: 92.119.112.114. (Knowing that IP, we could "pivot" on that IP address to potentially find other domains sharing that same IP address.)

- That IP also gets mapped using Oregon Routeviews data to the route "92.119.112.0/24", originated by AS204601

For details about who owns that ASN, you can check your favorite command line Whois client:

$ **whois AS204601**
[...]
aut-num: AS204601
as-name: ON-LINE-DATA
[...]
organisation: ORG-ZB24-RIPE
org-name: Zomro B.V.
country: NL
[...]

or check out <https://bgp.he.net/>). Once we know the ASN that originated an IP, we may also want to consider checking DNSDB for the other prefixes originated by that ASN, assuming we're still "on the hunt" for other domains of potential interest.

16. Running at Scale: Sending Our Remaining FQDNs Through DNSDB Standard Search

Now that you've seen how the process works, we then ran our whole set of just under 50,000 Flexible Search results through DNSDB Standard Search, once for "A" records and then a second time for "CNAME" records, just as we demonstrated in the previous section, but for all of our remaining FQDNs. Having done so, we ended up with:

- "A" Record (7 day time fence) results: 19,504 unique RRnames with a total of 21,663 results
- "CNAME" Records (7 day time fence) results: 3,471 unique RRnames with a total of 3,838 results
- Considering both "A" and "CNAME" results together, we have 22,964 unique RRnames. Eleven unique RRnames ((19504+3471)-22964=11) appear to have been seen with BOTH A records and CNAME records)

You may wonder:

- "So how do we end up having MORE results than unique RRnames?" The answer to that question is, "Some of the RRnames may have results from different bailiwicks, or may have results with different Rdata for the same RRname over the time fenced period."

When that happens, you'll get multiple results when you lookup a single (RRname, RRtype) combination in DNSDB.

- "How come ONLY 22,964 unique RRnames had results (out of the 49,945 we started with) when looking at a seven day time window?"

Well, this article was written in part over the winter holiday break, so some names that were seen just once during the initial seven day window didn't get seen again as our retrospective ("go-back-7 days") window "slid along."

We could have explicitly specified the same starting and ending times for our follow up DNSDB Standard Search runs as for our initial DNSDB Flexible Search run, but we decided to "take advantage" of the different windows — we're primarily interested in hits that WERE active and STILL ARE active, not "UFO" FQDNs that "blipped onto our radar" once, only to never be seen again.

17. Filtering Our DNSDB Standard Search Output: Ignoring Hits from Some ASNs

We're now able to filter more FQDNs by looking at our DNSDB Standard Search output. Let's begin by considering the ASNs we saw.

Assuming:

- The "A" record results from DNSDB Standard Search are in the file results.jsonl and
- The "CNAME" record results from DNSDB Standard Search are in the file result2.jsonl

We can extract those from our output files by saying:

```
$ cat results.jsonl results2.jsonl | jq '.dnsdbq_rdata[]?.asinfo.as[]' | more
13335
13335
13335
13335
13335
13335
13335
24940
24940
16807
[etc]
```

We find 1,603 unique ASNs via that approach. The 50 or so most-seen ASNs were:

$ cat results.jsonl results2.jsonl | jq '.dnsdbq_rdata[]?.asinfo.as[]' | sort | uniq -c | sort -nr > summary-dnsdbq-asns.txt

$ more summary-dnsdbq-asns.txt

```
   3270 16509      <-- Amazon
   2548 13335      <-- Cloudflare
   1812 4837       <-- China169-Backbone
   1427 15169      <-- Google
    994 37963      <-- Hangzhou Alibaba Advertising
    675 63949      <-- Linode
    668 58182      <-- Wix
    604 14618      <-- Amazon
    593 19574      <-- CSC (Corporation Service Company)
    503 46606      <-- Unified Layer
    435 16807      <-- IBM - Events Infrastructure
    408 24940      <-- Hetzner
    381 45090      <-- Shenzhen Tencent Computer Systems
    367 36351      <-- SoftLayer ("An IBM Company")
    339 8560       <-- Ionos (formerly "1&1 Internet SE")
    338 197695     <-- Reg.ru
    328 16276      <-- OVH
    306 53831      <-- Squarespace
    301 38283      <-- Chinanet SC Telecom
    286 14061      <-- Digital Ocean
    235 3320       <-- Deutsche Telekom
    232 17621      <-- China Unicom Shanghai
    220 40034      <-- Confluence Networks Inc, Tortola VG
    185 31624      <-- Verotel International BV, Amsterdam NL
    166 797        <-- AT&T Services, Enterprise IP Group
    157 48287      <-- JSC "RU-CENTER", Moscow
    152 47846      <-- Sedo
    151 40676      <-- Psychz Networks, Walnut Cal.
    150 22612      <-- Namecheap Inc.
    143 6724       <-- Strato AG
    136 9123       <-- TimeWeb Ltd, St Petersburg RU
```

```
129 198610    <-- Beget LLC, St Petersburg RU
126 16625     <-- Akamai
117 31034     <-- Aruba S.p.A., Italy
115 34788     <-- Neue Medien Muennich GmbH
115 32244     <-- Liquid Web, Lansing Mich.
115 19527     <-- Google
114 45102     <-- Alibaba (US) Technology Co
114 2635      <-- Automattic Inc, San Francisco
112 29873     <-- Newfold Digital Inc, Jacksonville FL
108 55990     <-- Huawei Cloud Service, Beijing
106 18779     <-- EGI Hosting, Santa Clara Cal.
 99 26496     <-- Godaddy
 97 54113     <-- Fastly
 96 25532     <-- Masterhost.ru, Moscow
 94 61969     <-- TeamInternet AG
 91 8075      <-- Microsoft
 90 134548    <-- DXTL Tseung Kwan O Service, HK
 86 19871     <-- Network Solutions LLC
 81 35369     <-- Linz Strom Gas Waerme GmbH, AT
 81 13886     <-- Cloud South, West Palm Beach FL
```

We're going to exclude (as being "directly IBM affiliated") anything originated by AS16807 ("IBM - Events Infrastructure"). That lets us exclude 431 hits:

$ **cat results.jsonl results2.jsonl | grep 16807 | jq -r '.rrname' | sort -u | reverse-domain-names**
[output from this command is available in Appendix II]

We're also going to exclude the 297 hits originated by 19574 ("CSC") since they tend to be extremely careful when it comes to their customers:

$ **cat results.jsonl results2.jsonl | grep 19574 | jq -r '.rrname' | sort -u | reverse-domain-names**
[output is available in Appendix III]

While we're excluding things, there are other IBM ASNs we might also consider excluding.

Checking a list of known "IBM" ASNs against the ASNs we saw in our results, let's also ignore any hits associated with:

ASN	ASN Name
19603	IBM
27797	IBM Argentina S.R.L
43719	IBM Services Financial Sector Luxembourg Sarl
61179	IBM Romania SRL

$ **cat results.jsonl results2.jsonl | egrep -v 'as":\[(16807|19574|19603|27797|43719|61179)\]' > after-asn-filtering.jsonl**
$ **wc -l after-asn-filtering.jsonl**
24759 after-asn-filtering.jsonl

18. Excluding RRsets by "Trimming the Lower Tail" Based on the Counts Reported by DNSDB Standard Search

If we rarely see something, it's unlikely to be of substantial ongoing interest. We now have count data for each unique RRset. How many results could we exclude if we picked a particular low-count threshold? Let's try some discrete values and see how many observations are at-or-below that threshold. We could do this value-at-a-time, by running a series of commands like:

$ **jq -c 'select(.count <= 1)' after-asn-filtering.jsonl | wc -l**
3745
$ **jq -c 'select(.count <= 2)' after-asn-filtering.jsonl | wc -l**
4849
$ **jq -c 'select(.count <= 2)' after-asn-filtering.jsonl | wc -l**
5370

That demonstrates that:

- 3,745 of our 24,759 residual hits have only been seen once
- 4,849 out of 24,759 have been seen once or twice, and
- 5,370 have been seen 1 to 3 times.

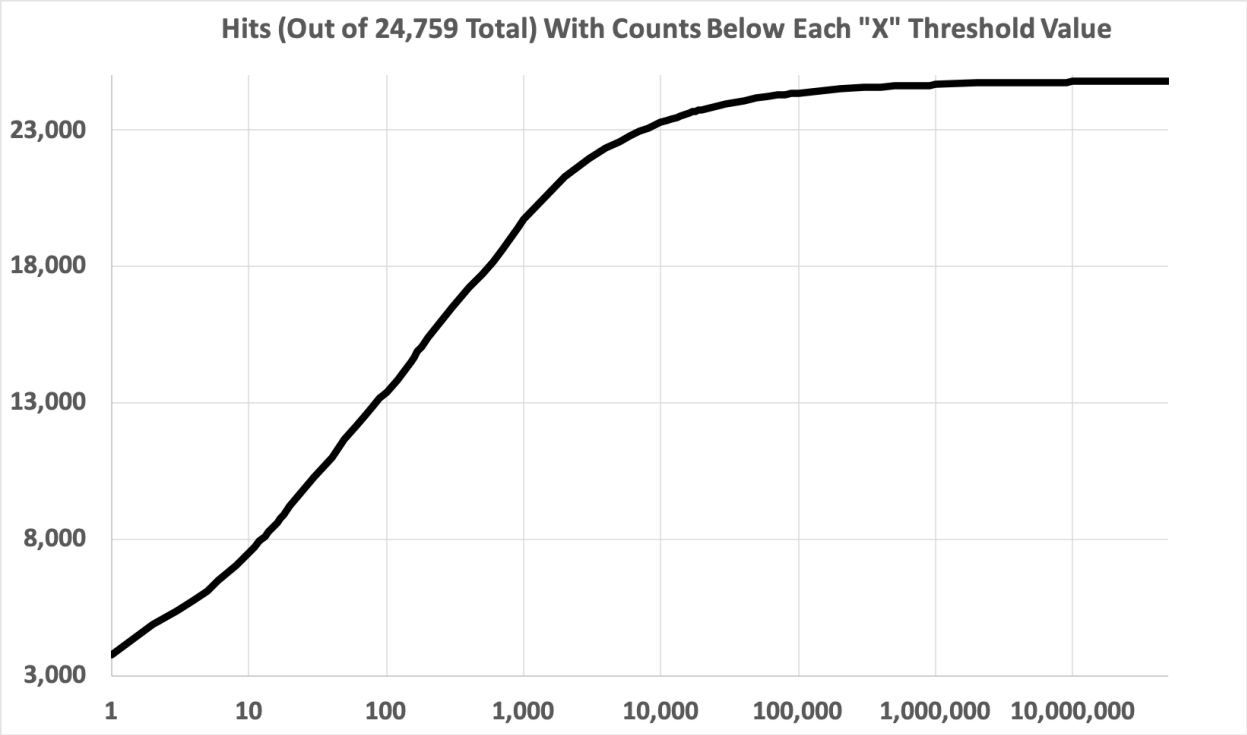
We could obviously proceed to do similar probes of whatever potential threshold values we might like.

Rather than running that sort of command manually a bunch of times, though, let's just use a little bash script to test a set of selected "cut points:"

$ **cat check-cut-points.bash**
```
for i in 1 2 3 4 5 6 7 8 9 10 \
        11 12 13 14 15 16 17 18 19 20 \
        30 40 50 60 70 80 90 100 \
        120 130 140 150 160 170 180 190 200 \
```

```
        300 400 500 600 700 800 900 1000 \
        2000 3000 4000 5000 6000 7000 8000 9000 10000 \
        11000 12000 13000 14000 15000 16000 17000 18000 19000 20000 \
        30000 40000 50000 60000 70000 80000 90000 100000 \
        200000 300000 400000 500000 600000 700000 800000 900000 1000000 \
        2000000 3000000 4000000 5000000 6000000 7000000 8000000 9000000 \
        10000000 20000000 30000000 40000000 50000000
do
  echo -n "$i "
  jq -c "select(.count <= $i)" after-asn-filtering.jsonl | wc -l
done
```

After running that script, we'll graph the output from that script (note the logged X axis)! That graph looks like:



We're going to exercise our professional discretion and decide to exclude any hits that were seen only three times or less on the low end, regardless of any other considerations:

$ jq -c 'select(.count > 3)' after-asn-filtering.jsonl > after-asn-filtering-greater-than-3.jsonl
$ wc -l after-asn-filtering-greater-than-3.jsonl
19389 after-asn-filtering-greater-than-3.jsonl

19. Excluding RRsets by "Trimming (Some Of) the Upper Tail" Based on the Counts Reported by DNSDB Standard Search

Looking further at the previous graph, clearly there are MANY observations that had been seen only a handful of times, but there were also some observations that have been seen awfully often. These domains might be considered as making up the "right hand" or "upper" tail of our distribution. Just as we might not be very interested in things that are only seen rarely, we may also not be particularly interested in studying routine things that are seemingly seen "all the time."

For example, let's look at the observations that have counts in excess of 40,000,000. We'll show those RRnames in reversed format with a blank line added between hits to help improve readability of the following:

$ jq -c 'select(.count > 40000000)' after-asn-filtering.jsonl

{"count":43075515,"time_first":"2013-09-29 23:52:23","time_last":"2022-01-04 04:26:58",**"rrname":"ru.sibmama.ns1"**,"rrtype":"A",**"bailiwick":"ru."**,"rdata":["46.61.143.6"],"dnsdbq_rdata":{"46.61.143.6":{"asinfo":{"as":[12389],"cidr":"46.61.128.0/17"}}}}

{"count":43191703,"time_first":"2013-09-29 21:12:20","time_last":"2022-01-04 02:00:12",**"rrname":"ru.sibmama.ns1"**,"rrtype":"A",**"bailiwick":"sibmama[dot]ru."**,"rdata":["46.61.143.6"],"dnsdbq_rdata":{"46.61.143.6":{"asinfo":{"as":[12389],"cidr":"46.61.128.0/17"}}}}

{"count":44805915,"time_first":"2013-11-01 14:47:37","time_last":"2022-01-04 02:34:39",**"rrname":"ru.sibmama"**,"rrtype":"A",**"bailiwick":"sibmama[dot]ru."**,"rdata":["46.61.143.6"],"dnsdbq_rdata":{"46.61.143.6":{"asinfo":{"as":[12389],"cidr":"46.61.128.0/17"}}}}

Glancing casually at those results, you might think that there must be something wrong — could we actually be seeing the "same result" three times? No. Those are three distinct and unique results, although they admittedly look quite similar if you just casually glance at them, particularly if you look at them in raw format (without the highlighting we've manually added above).

If we do scrutinize those manually highlighted bits, we can the differences between the three hits:

- The last of those three results is for the base domain (sibmama[dot]ru). The company's sensors saw 44,805,915 cache misses for that name's "A" record, all pointing at the IP address 46.61.143.6.
- The other two hits are for the name server domain ns1.sibmama[dot]ru (which is hosted on the same IP address as the base domain name).

The difference between the two hits for ns1.sibmama[dot]ru is **the bailiwick** (or "where in the DNS hierarchy this data comes from"). For more on bailiwicks: <https://www.farsightsecurity.com/blog/txt-record/what-is-a-bailiwick-20170321/>

In this example, one of the results comes from the TLD (e.g., dot ru), while the other one comes from the delegation point (e.g., the sibmama[dot]ru domain itself).

Substantively, if we visit the named Russian-language site from a test/lab system, there's nothing particularly noteworthy/"ibm"-related about it. We can safely ignore it. Now let's look at hits with counts of at least a million or more:

$ **jq -c 'select(.count >= 1000000)' after-asn-filtering-greater-than-3.jsonl | jq -r '.rrname' | sort**

We see reversed-format domains related to three categories of domains:

a) Domains that are known/likely-to-be IBM-controlled or affiliated. These may not have been noticed in previous "ibm ASN" screens because these domains may be leveraging a cloud service. We're going to exclude those as presumptively "already known" to the brand owner.

com.box.ent.ibm	<-- likely actually IBM-related
com.espn.hosted.geo.ibm-fantasy-widget	<-- likely actually IBM-related
com.ibm	<-- IBM-related
com.ibmmarketingcloud.api1	<-- Centerbridge Partners acquired IBM's marketing and commerce software offerings in July 2019
com.ibmmarketingcloud.api2	
com.ibmmarketingcloud.api3	
com.ibmmarketingcloud.api4	
com.ibmmarketingcloud.api5	
com.ibmmarketingcloud.transact3	
com.ibmmarketingcloud.transact4	
com.ibmmarketingcloud.ubx.api-01	
com.ibmmarketplace.myibm	<-- IBM-owned domain
com.ibmserviceengage	<-- IBM-owned domain
com.ibmserviceengage.static	
com.ibmserviceengage.static	
com.ihost.usf.fr2.fribmmop2nmxdns-01	<-- ihost.com is an IBM-owned domain
com.ihost.usf.fr2.fribmmop4nmxdns-01	
com.seismic.ibm	<-- auth-protected page, apparently IBM-related
com.weather.redirector-prod-dal10-ibm	<-- weather.com is an IBM business
com.weather.redirector-prod-dal12-ibm	
com.weather.redirector-prod-fra02-ibm	
com.weather.redirector-prod-fra02-ibm	
com.weather.redirector-prod-fra05-ibm	

com.weather.redirector-prod-seo01-ibm
com.weather.redirector-prod-wdc04-ibm
com.weather.redirector-prod-wdc06-ibm
com.webex.ibm <-- Webex@IBM
com.webex.ibm2 <-- Webex@IBM
ibm.nic.a <-- IBM's own TLD
ibm.nic.b
ibm.nic.c
ibm.nic.d
net.cs186.com.ibm.www <-- on a site that has an IBM multidomain TLS cert
net.edgekey.com.ibm <-- assumed to be legitimately IBM-related
net.edgekey.com.ibm.commercelibs
net.edgekey.com.ibm.developer

net.edgekey.ibmcom.outer-ccdn-dual
net.edgekey.ibmcom.outer-global-dual
Net.edgekey.ibmcom.outer-global-v4

b) Domains that appear to be composed from multiple concatenated words (or word-segments), coincidentally forming "ibm" at the point of conjunction. We're going to exclude these as having a low probability of being maliciously targeted:

com.habibmetro.metromail <-- ("habib" "metro")
net.duba.libmini <-- ("lib" "mini")
net.omtrdc.insight.usbankribmetrics <-- ("rib" "metrics")
org.bibme.ads <-- ("bib" "me")
org.bibme.support
org.oclc.contentdm.libmma <-- ("lib" "mma")
ru.distribmail.goappsdl <-- ("distrib" "mail")
ru.sibmama <-- ("sib" "mama")
ru.sibmama.blog
ru.sibmama.club
ru.sibmama.dom
ru.sibmama.forum
ru.sibmama.foto
ru.sibmama.line
ru.sibmama.ns1
ru.sibmama.r
ru.sibmediafon.ns <-- ("sib" "media")
ru.sibmediafon.ns1
ru.sibmes.ns1 <-- ("sib" "mes" ?)
ru.sibmes.ns2

c) Our third and final category consists of domains of unknown status. We would NOT suggest pre-emptively excluding these:

com.btconnect.ibmr
com.dynapis.rum.ibm-ams1
com.dynapis.rum.ibm-iad1
com.dynapis.rum.ibm-sjc1
com.ibm-pc.ns1
com.ibm-pc.ns2
com.ibmcy.y
com.ibmoto.erebus
com.manulife.cgefaibmglbp1
com.manulife.clefaibmglbp1
com.mediaroom.ibmnews
com.sendibm1.d.a.img
com.sendibm1.d.a.r
com.sendibm1.r
com.sendibm3.d.ag.img
com.sendibm3.d.ag.r
com.sendibm4.d.ah.img
com.sendibm4.d.ah.r
com.service-now.ibmaabpr
com.sibmail
com.sibmail.www
com.spdydns.net.duba.libmini
com.wibmo.hdfc-acs
eu.ibmhosting.ns3
eu.medallia.digital-cloud-ibm
eu.medallia.digital-cloud-ibm.resources
net.demdex.ibm
net.yandex.cdn.cache-novosibmgf01
net.yandex.cdn.cache-novosibmgf02
net.yandex.cdn.cache-novosibmgf03
net.yandex.cdn.cache-novosibmgf04
net.yandex.cdn.cache-novosibmgf05
net.yandex.cdn.cache-novosibmgf06
net.yandex.cdn.cache-novosibmts01
net.yandex.cdn.cache-novosibmts02
net.yandex.cdn.cache-novosibmts03
net.yandex.cdn.cache-novosibmts04
net.yandex.cdn.cache-novosibmts05
net.yandex.cdn.cache-novosibmts06
net.yandex.strm.ext-strm-novosibmgf01

net.yandex.strm.ext-strm-novosibmgf02
net.yandex.strm.ext-strm-novosibmgf03
net.yandex.strm.ext-strm-novosibmgf04
net.yandex.strm.ext-strm-novosibmgf05
net.yandex.strm.ext-strm-novosibmgf06
net.yandex.strm.strm-novosibmts01
net.yandex.strm.strm-novosibmts02
net.yandex.strm.strm-novosibmts03
net.yandex.strm.strm-novosibmts04
net.yandex.strm.strm-novosibmts05
net.yandex.strm.strm-novosibmts06
pe.net.ibm.dns1
pe.net.ibm.dns2

We'll exclude the a) and b) group data with:

$ **grep -v -f excludeables.txt after-asn-filtering-greater-than-3.jsonl > after-asn-filtering-and-filtering-some-millions.txt**
$ **wc -l after-asn-filtering-and-filtering-some-millions.txt**
19203 after-asn-filtering-and-filtering-some-millions.txt

20. Looking at Rdata IP Address Data ASNs (Weighted by DNSDB Count Data)

Now let's focus on the actual rdata we've received from DNSDB Standard Search. We'll extract that data (and the associated ASN info and count data) for all our remaining hits:

$ **jq -r '"\(.dnsdbq_rdata[]?.asinfo.as[]) \(.rdata[]) \(.count)"' < after-asn-filtering-and-filtering-some-millions.txt | sort -n > rdata-after-asn-filtering-and-filtering-some-millions-sorted.txt**
$ **wc -l rdata-after-asn-filtering-and-filtering-some-millions-sorted.txt**
59195 rdata-after-asn-filtering-and-filtering-some-millions-sorted.txt

Some of that output file looks like the following (each line has the ASN, IP and count for that observation):

[...]
18 146.6.139.178 704
29 128.36.209.32 265
29 130.132.16.239 244
29 130.132.16.239 319
29 130.132.16.3 356
32 **10.39.20.133** 817
32 **10.39.20.55** 852
32 **10.39.20.55** 852

[...]

It's pretty hard to miss that some of those IPs are from "RFC1918 private address space" (in this case, note that the highlighted addresses are from the private address range 10.0.0.0/8). We could go through and manually filter those out, but let's take a more overarching approach and see if we can answer the question, *"What ASN do we see used most heavily?"*

Conceptually, that means we want to sort by ASN, sum the counts within each ASN, and then sort those aggregated counts in descending order. Rather than doing this with Un*x command line utilities, let's use GNU PSPP (<https://www.gnu.org/software/pspp/>), a free/open source (and largely syntax-compatible work-alike) version of a popular commercial statistical package. The code needed to find the top ASNs is pretty short:

```
data list
  file='rdata-after-asn-filtering-and-filtering-some-millions-sorted.txt'
  free / asn * ipaddress (a15) count *
print formats count (f10.0) asn (f10.0)
sort cases by asn ipaddress
aggregate outfile=*
  / presorted
  / break=asn
  / asn_total = sum(count)
print formats asn_total (f14.0)
sort cases asn_total (d) asn
list asn_total asn
```

We can then run that code by saying:

$ **pspp < read-and-summarize.psp > read-and-summarize.output**

An excerpt of the output from that run looks like:

```
+---------+------+
|asn_total|  asn |
+---------+------+
| 64164335| 13335|        <-- Cloudflare
| 33717604| 16509|        <-- Amazon
| 30788292| 13238|        <-- Yandex Russia
| 30179791| 14618|        <-- Amazon
| 25021716|  2856|        <-- BT
| 17228960| 36351|        <-- Softlayer (an IBM company)
| 12999113| 49505|        <-- Selectel Russia
| 12805416| 31133|        <-- PJSC MegaFon Russia
| 10817391| 56981|        <-- CJSC "ER-Telecom Holding" Tomsk branch Russia
```

| 9663786| 39337| <-- JSC "Corp Soft" Russia
| 9115373|135751| <-- Enstage Software Pvt Ltd India
| 7276648| 4249| <-- Eli Lilly
| 7035197| 54915| <-- Manufacturers Life Insurance Canada
| 6960028| 54916| <-- Manufacturers Life Insurance Canada
| 6956414| 12252| <-- America Movil Peru
| 6917432|208722| <-- Yandex Finland
| 6774974| 13916| <-- Proofpoint
| 6443084| 47764| <-- Mail.ru Russia
| 4197748| 14870| <-- Flexera Software
| 3900771| 14061| <-- Digital Ocean
| 3729215| 17225| <-- AT&T Enhanced Network Services
| 3116934| 17227| <-- AT&T Enhanced Network Services
| 2876979| 5429| <-- UMOS Russia
| 2186551| 16839| <-- SERVICENOW
| 2095811| 33011| <-- Box.com
| 2037141| 24940| <-- Hetzner Germany
| 1813240| 4837| <-- China Unicom China169
| 1735553| 7018| <-- AT&T Services Inc
| 1717125| 50673| <-- Serverius Netherlands
| 1477433| 2635| <-- Automattic, Inc
| 1460162| 19795| <-- Acoustic, L.P.
| 1379814| 1221| <-- Telstra
| 1378539| 12389| <-- ROSTELECOM Russia
| 1370726| 2686| <-- AT&T Global Network Services
| 1341210|396982| <-- Google
| 1247400| 20141| <-- Quality Technology Services
| 1223761| 55855| <-- PLAYPARK PTE LTD Singapore
| 1155195| 3209| <-- Vodafone Germany
| 1082585|197695| <-- Reg.ru Russia
| 1081587| 3598| <-- Microsoft
| 1081587| 3598| <-- Microsoft
| 1013597| 3320| <-- DT Germany
| 1009564|198610| <-- Beget LLC Russia
[all remaining have a count < 1,000,000]

Some of the entities on that list may be cloud service providers offering "zero-human-interaction-required" insta-provisioning, which may increase the likelihood that virtually anonymous (and potentially malicious) customers may be leveraging their infrastructure. There have been some Federal efforts to explicitly encourage cloud provider "know your customer" ("KYC") policies, such as U.S. Executive Order 13984 of January 19, 2021, "Taking Additional Steps to Address the National Emergency with Respect to Significant Malicious Cyber-Enabled Activities," <https://www.govinfo.gov/content/pkg/FR-2021-09-24/pdf/2021-20430.pdf>, but nothing cybersecurity-related happens "overnight" (and obviously US regulations have limited applicability abroad).

In other cases, a single domain may drive virtually all of a given ASN's "ibm"-related hits. For example, consider AS55855 ("PLAYPARK PTE LTD Singapore"). We can see that a single "ibm"-related domain (and associated hosts thereunder) is the source of all the post-screening "ibm" hits we found for that ASN. Check out the following (reversed-label format):

$ **cat results.jsonl results2.jsonl | fgrep "[55855]" | jq -r '.rrname' | sort -u**
net.cibmall
net.cibmall.mat
net.cibmall.fbapps
net.cibmall.guild-mat
net.cibmall.yulgang
net.cibmall.payment
net.cibmall.home
net.cibmall.images
net.cibmall.bbv2
net.cibmall.gj
net.cibmall.register
net.cibmall.xtl
net.cibmall.mh
net.cibmall.bbs
net.cibmall.sdox
net.cibmall.forum
net.cibmall.www
net.cibmall.event
net.cibmall.tlbb
net.cibmall.mat2
net.cibmall.members

Another pattern we may see when we drill down may be sets of domains that look as if they may have been algorithmically generated, such as the following (again, note the label-reversed format):

$ **cat results.jsonl results2.jsonl | fgrep "[14618]" | jq -r '.rrname' | sort -u**
[...]
info.aewktizibmrxsodmhupfwkbymf
info.ampfwkobgibmwgfyhuwrwvcttt
info.bmmvaueiylvstcfejnljibmfcd
info.caddlzdhswizzknprhibmbytmf
info.cecunzdvrfswibmhuijfup
info.cutyhtgydyibmvxgpdxcey
info.dbybmibmjlrdutzfyxfiojhmbq
info.dexnbqgayveibmdbijvhjkn
info.eibmfiydkbydorspqwbuip

info.eypzpltopfzxaibmembyxtjflb
info.fqlkveuyceznfhquibmzuqgwcso
info.habeemgvadbeeorbibmypdipvyp
info.hbfuingibmhxkbmlifdmjzba
info.hdydifhmwsswtzbiuconibmj
info.ibydnzlcibmpfytbqjnuoiblz
info.ifibmzhvwijvhytqchmhciswobgy
info.kfxibmrhbyuevo
info.lbfuiljvonhircibmbtgjnmvcq
info.mflvvkthbmvtqibmzdexogiei
info.mnguzxjqsxcusxnzgyibmfcehugd
info.mucfmdknytoflvemecibmpzlzfuby
info.nbyylmfpmvmribmnwgcaylifbau
info.nforzltkzheypvhibmttkswijxg
info.pbgmxskzuccybmheibmbzdojhmbyhz
info.pfemfacibmizcatjobaikjxprjzto
info.pjdibmuwwqajnjrolfdtgjnh
info.pnhuorsgibmjqobvfyroovjfdrgjf
info.povibmbfuhuqkpblnizxdxlxs
info.prlribmfzxltlfdbiaeursdemzd
info.qmpojvoaulribmvpzpzdif
info.swqhgmrkjvwdqjzxodudibmfzlhy
info.tknfibmrancmvqklbxofugiz
info.tsztxxhibmjvbamvlnypbtceywk
info.ukxcdibmztxtojifkrbmlbem
info.uqojbqkfmusinfekjhibmeqkvgdu
info.vcshqpcqbibmxctrkdiwsdda
info.vgtfemndctibmnbenvxrkobzp
info.vhibmlpcqaithacilfxoizbicu
info.wofqgahhibmkzpnjo
info.xfqgibmqsxbijvemtsbqhatvsdmpn
info.xofxibmfwjvlpnxcjrtppnqsby
info.ycibmrocyknztfyhutfytuwgur
info.ylxofmbthxibmfmlaqxknq
info.ytnfjfscdhatxlkdibmxauynr
info.ytwahibmfetgmzxhyvgcfqojpnr
info.ztlepndeacqnnfeaculhibm
info.zxpytkzlnkztnjdibmbiorikfnv
[...]

Those apparently algorithmically-generated names may not be of interest to brand protection staff, but may be of interest to other cyber researchers (in this case, additional similarly-algorithmic-looking domains are able to be found by pivoting on the nameserver ns1.kratosdns[dot]net)

Phase III: Domain Reputation Phase

21. Domain Reputation

Historically, Farsight Security has NOT done domain reputation analysis, but DomainTools (which recently acquired Farsight), does offer this, see <https://www.domaintools.com/resources/api-documentation/reputation>

Let's see how the reputation of our residual domains looks when run through the DomainTools reputation engine. There are many ways one could do those checks, but for consistency with the rest of this analysis, let's use the basic command line domaintools client that's available via <https://github.com/DomainTools/python_api> or via pip3.

We can install that package with pip3 by saying:

$ **pip3 install domaintools_api --upgrade**

Having done so, we'll then use a text editor to put our API username and API key into ~/.dtapi (your username should be entered on the first line of that file, and your API key should be entered on the second line). [If you're licensed to use the DomainTools Reputation API, but don't remember your DomainTools username or API, see <https://account.domaintools.com/api/dashboard/>]

Ensure that the file containing your credentials is not accessible by anyone else who may also be using your system:

$ **chmod 400 ~/.dtapi**
$ **ls -la ~/.dtapi**
-r-------- 1 jsmith staff 39 Jan 9 12:03 /Users/jsmith/.dtapi

Once you've installed the domaintools_api package and setup your credentials, you can then try doing a sample run by saying:

$ **domaintools reputation www.google.com**

```
{
        "response": {
        "domain": "google.com",
        "risk_score": 0
        }
}
```

If you want to dig into WHY a domain has the score it does, you can try running the domaintools command with the risk_evidence option instead of the reputation option:

$ **domaintools risk_evidence www.google.com**


```
{
        "response": {
        "domain": "google.com",
        "risk_score": 0,
        "components": [
        {
        "name": "zerolist",
        "risk_score": 0
        }
        ]
        }
}
```


If we don't want JSON output and just want to see the reputation score and domain name as "plain text", you can use jq to get that output:

$ **domaintools reputation google.com | jq -r '"\(.response.risk_score) \(.response.domain)"'**
0 google.com

Since that approach works for one domain, we can repeat that process for each of the residual domains we're curious about.

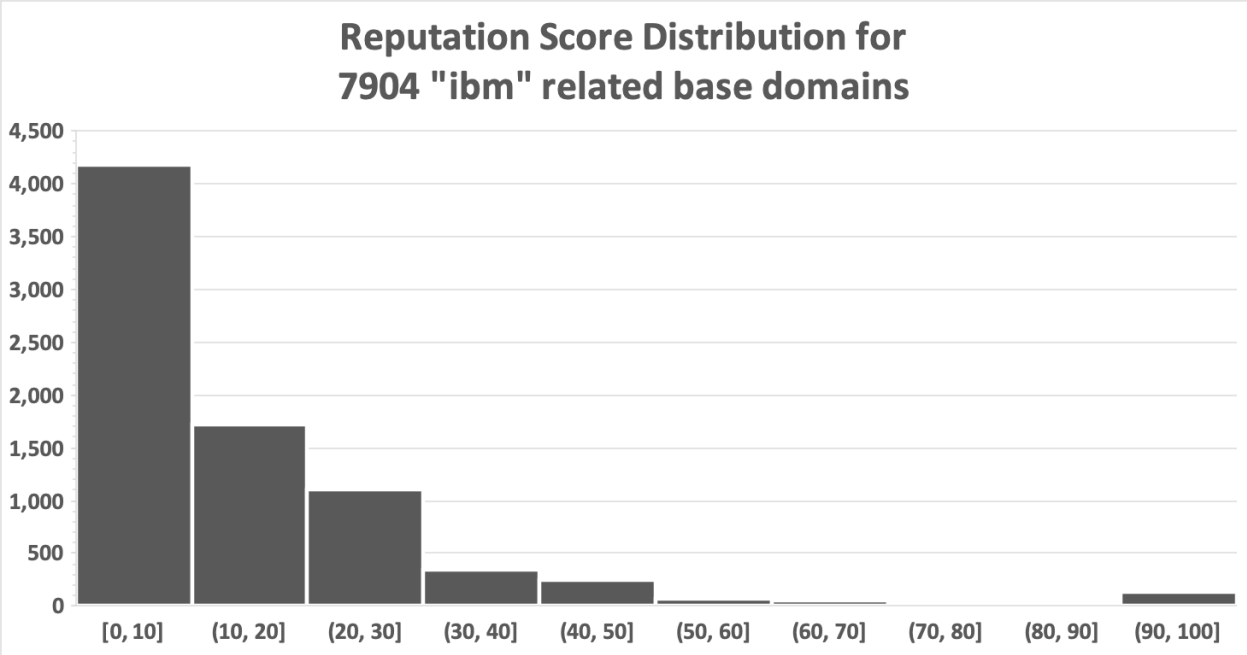
Use your favorite editor to create a file called doms-to-test.bash that looks like (note substitution of [dot] for "real dots"):

```
domaintools reputation 01-ibm[dot]com | jq -r '"\(.response.risk_score) \(.response.domain)"'
domaintools reputation 0123456789[dot]tw | jq -r '"\(.response.risk_score) \(.response.domain)"'
domaintools reputation 01finance[dot]vip | jq -r '"\(.response.risk_score) \(.response.domain)"'
[etc]
```

Run that simplistic "script" by saying:

$ **chmod u+rx doms-to-test.bash**
$ **bash doms-to-test.bash | sort -u > doms-to-test.output**
$ **wc -l doms-to-test.output**
7904 doms-to-test.output

After that script runs, we can sort that file and graph the resulting distribution of reputation scores as a histogram in Excel or another graphing program of your choice:



Clearly, many of the domains we've found have a very low-to-moderate risk (where a score of 0=least risk), but there are some domains we've found that have a score of 90-100, maximal (or near-maximal) risk scores. If we wanted to, we could select just the domains that have a score of 90 or higher by saying:

$ **awk '$1 >= 90' < doms-to-test.output | sort -n > doms-to-test-2.output**
$ **wc -l doms-to-test-2.output**
138 doms-to-test-2.output

To avoid accidentally triggering spam/malware filtering of this report, we're going to refrain from showing those selected highest score domains here, but obviously those worst-scoring domains would be the ones we'd prioritize for further in-depth investigation or action if we had limited resources.

Abbreviated DomainTools Reputation API Q&A:

Question 1): "Is there a command summary available for the domaintools command line interface?"
Answer: Use $ **domaintools --help**

Question 2): "How can I see what my DomainTools Reputation API quota is, and how much of my quota I've used?"
Answer: Use $ **domaintools account_information**

Question 3): "Why not just pull reputation scores for ALL the FQDNS we originally uncovered?"
Answer: Most DT API users have a finite DomainTools query quota (just as most DNSDB API users do). Running all the domains you may potentially have uncovered may exceed your query quota if you elect to forgo initial screening.

Question: 4) "Is individualized reputation data available for each FQDN or is reputation done only for "base domain names"?

Answer: Reputation is reported for the base domain names. As stated in the DomainTools reputation API documentation:

Note that if you provide a hostname (e.g. www.domaintools.com) rather than a domain (e.g. domaintools.com) we will attempt to return the risk score for the domain, and the domain we used to lookup the risk score will always be returned in the response.

Thus, if you supply a FQDN rather than just a base domain name, the DomainTools reputation API will deduce the base domain name for the FQDN you've entered and correctly return reputation data for that base domain. However, it would be a waste of your DomainTools API reputation query quota to check the reputation for multiple FQDNs that are all just variants of the same base domain since they'd all return the same reputation score.

Conclusion

22. Wrapping It All Up

You've now seen a walkthrough of what can be involved with attempts to find unusually short or common patterns (such as "ibm"), in DNSDB Flexible Search and DNSDB Standard Search. We've also shown you how you can use the DomainTools Reputation API to get a risk score for domain names of interest.

Along the way, you've also learned a little about some Un*x commands, scripting, and regular expressions.

You've also been exposed to the issues that can arise when working with short or common patterns, including:

- Finding a potentially overwhelming number of results (potentially four million or more!)
- Getting an incomplete set of results if your query is too general or your time fence is too wide
- Wildcard domain-related "noise" issues
- Rarely-seen FQDNs "aging" out of a short time fence (if you're using relative time fencing and your initial Flexible Search discoveries and your follow-up Standard Search queries aren't cotemporaneous)
- More results than you might expect from DNSDB Standard Search due to changing Rdata during the timefenced period, and/or bailiwick driven effects

When you begin to work with more reasonable-length patterns, you may be surprised by how easy that is in comparison.

Appendix I. 1st-level-dom, 2nd-level-dom, and reverse-domain-names scripts

$ **cat 1st-level-dom**

```
#!/usr/bin/perl
use strict;
use warnings;
use IO::Socket::SSL::PublicSuffix;

my $pslfile = '/usr/local/share/public_suffix_list.dat';
my $ps = IO::Socket::SSL::PublicSuffix->from_file($pslfile);

my $line;

foreach $line (<>) {
        chomp($line);
        my $root_domain = $ps->public_suffix($line,0);
        printf( "%s\n", $root_domain );
}
```

$ **cat 2nd-level-dom**

```
#!/usr/bin/perl
use strict;
use warnings;
use IO::Socket::SSL::PublicSuffix;

my $pslfile = '/usr/local/share/public_suffix_list.dat';
my $ps = IO::Socket::SSL::PublicSuffix->from_file($pslfile);

my $line;

foreach $line (<>) {
        chomp($line);
        my $root_domain = $ps->public_suffix($line,1);
        printf( "%s\n", $root_domain );
}
```

$ **cat reverse-domain-names**

```
#!/usr/bin/perl

my @lines = <>;
chomp @lines;

@lines =
        map { join ".", reverse split /\./ }
        sort
#   map { join ".", reverse split /\./ }
        @lines;

print "$_\n" for @lines;
```

Appendix II. List of IBM-related FQDNs hosted in AS16807 ("IBM - Event Infrastructure")

ibm.am
ibm.asia
ibm.co.at
ibm.at
ibm.com.au
ibm.net.au
ibm.be
ibm.bi
ibm100.biz
ibmparts.biz
ibmwatsonhealth.biz
ibmwatson.blue
ibm.com.br
ibmbrasil100.com.br
ibm.ca
ibmstore.ca
ibmwatson.ca
ibm.career
ibmwatson.careers
ibm.ceo
ibm.ch
ibm.cl
ibmthink.cl
ibm.cloud
ibmopenshift.ibm.cloud
ibmevents.cloud

ibmwatson.club
ibm.com.cn
ibm.cn
ca.ibm.cn
cn.ibm.cn
ibm.com.co
ibmcapital.co
01-ibm.com
03-ibm.com
10ibm.com
304-ibm.com
520ibm.com
aeibm.com
anibm.com
aoibm.com
beibm.com
bfibm.com
bgibm.com
bhibm.com
boibm.com
bsibm.com
career-hr-ibm.com
hostmaster.career-ibm.com
careeribm.com
cgibm.com
citizenibm.com
cloud-ibm.com
coibm.com
comibm.com
default._bimi.comibm.com
ibm.comibm.com
contactibm.com
cribm.com
cwibm.com
cyibm.com
czibm.com
dearibmboard.com
demoibm.com
dkibm.com
dzibm.com
eeibm.com
egibm.com
esibm.com

fishkillibm.com
foreverloveibm.com
ghibm.com
gribm.com
helloibm.com
hnibm.com
hribm.com
huatongibm.com
huibm.com
hursleyibm.com
ibm-100.com
ibm-at-100.com
ibm-cloud-innovation.com
ibm-db2.com
ibm-hpc.com
ibm-invest.com
ibm-iot-blog.com
ibm-latam.com
ibm-p.com
ibm-promotions.com
ibm-ready.com
ibm-sap.com
ibm-smart-cloud.com
ibm-think.com
ibm-usa.com
ibm-watson-for-engineering.com
ibm1860.com
ibm2025.com
ibm369.com
ibm370.com
ibm66.com
ibmadvantage.com
ibmanalyticsservice.com
ibmassist.com
ibmat100.com
ibmbcs.com
ibmbh.com
ibmblockchainaccelerator.com
ibmblr.com
ibmbluecare.com
ibmbluedirect.com
ibmcareer.com
ibmchampions.com

ibmchess.com
ibmclarity.com
ibmclinicaldevelopment.com
ibmcloud.com
ibmcloudmonitor.com
ibmcloudprivate.com
ibmclub.com
ibmcollabcloud.com
ibmcollege.com
ibmcompany.com
ibmconnectionsblog.com
ibmconsulting.com
ibmconsumerchannel.com
ibmcorporation.com
ibmcreditcard.com
ibmdatabasemag.com
ibmdatarecovery.com
ibmdatasystems.com
ibmdelivers.com
ibmdesign.com
ibmdev.com
ibmdeveloperday.com
ibmdsseries.com
ibmecmblog.com
ibmedge.com
ibmemployeebenefit.com
ibmemployeebenfits.com
ibmemployeebenifits.com
ibmfacts.com
ibmfilenetexperts.com
ibmfood.com
ibmfuturist.com
ibmglobalentrepreneur.com
hostmaster.ibmglobalentrepreneur.com
ibmglobalservices.com
ibmgroup.com
ibmguide.com
ibmhealthcare.com
ibmhp.com
ibminfonet.com
ibminteractive.com
ibmip.com
ibmiproposal.com

ibmiproposals.com
ibmit.com
ibmix.com
ibmjobs.com
ibmlotusprotector.com
ibmmainframe.com
ibmmainframeguru.com
smtp.ibmmainframeguru.com
ibmmanwithmachine.com
ibmmarketplace.com
ibmmaximo.com
ibmmobilefirstprotect.com
ibmn.com
ibmnegotiators.com
ibmoffice.com
ibmonlinestore.com
ibmparty.com
ibmpathways.com
ibmpcgroup.com
ibmpensionstrust.com
ibmpensiontrust.com
ibmpolicy.com
ibmpop.com
ibmpowerhour.com
ibmprivacy.com
ibmquantumawards.com
ibmreferrals.com
ibmresearch.com
ibmsecurity.com
ibmserverparts.com
ibmservicegroup.com
ibmsmart.com
ibmsmartbusinesscloud.com
ibmsmartercommerce.com
ibmsoftware.com
ibmsort.com
ibmsports.com
ibmstoragessa.com
ibmstore.com
ibmsupport.com
cpanel.ibmsupport.com
webdisk.ibmsupport.com
ibmtapesolutions.com

ibmtechnologistas.com
ibmteknoloji.com
ibmtelecoms.com
ibmthink.com
ibmtraining.com
ibmtransparency.com
ibmuc2.com
ibmucsquared.com
ibmupdater.com
ibmupdateservices.com
ibmvr.com
ibmwatson.com
ibmwatson-engineering.com
ibmwatsonecosystem.com
ibmwatsonhealth.com
ibmwatsontrend.com
ftp.ibmwatsontrend.com
ibmwatsonworks.com
ibmwatsonworkspace.com
ibmweb.com
ibmwebdeveloper.com
cpanel.ibmwebdeveloper.com
in.ibmwebdeveloper.com
kartikeyaenterprise.ibmwebdeveloper.com
ibmwebinar.com
ibmworkspace.com
ibmxserver.com
hostmaster.ibmxserver.com
ibmzjobs.com
ibmzsort.com
ibmzthemovie.com
ieibm.com
iibms.com
jmibm.com
job-ibm.com
jobs-ibm.com
jpibm.com
keibm.com
kribm.com
kwibm.com
laptopibm.com
lkibm.com
ltibm.com

lvibm.com
madewithibm.com
maibm.com
mgibm.com
mkibm.com
ibmxforce.mkt7665.com
ibmxforce.mkt7666.com
ibmcommerce.mkt7730.com
moibm.com
muibm.com
mwibm.com
ngibm.com
notesdevibm.com
nzibm.com
omibm.com
peibm.com
phibm.com
planetibm.com
plibm.com
ptibm.com
pyibm.com
qaibm.com
quanten-ibm.com
recruits-ibm.com
redhatibm.com
researchibm.com
roibm.com
saibm.com
sgibm.com
shop-ibm.com
shopibm.com
siibm.com
slibm.com
snibm.com
sortibm.com
support-ibm.com
sz-ibm.com
tdibm.com
team-ibm.com
thinkibm.com
thinkipibm.com
tnibm.com
uaibm.com

ugibm.com
veibm.com
vnibm.com
www-ibm.com
zurichibm.com
ibmwatson.community
ibmwatson.courses
ibm.cz
ibm.de
ibm.dk
ibmss.education
ibmwatson.education
ibm.es
ibm.eu
ibmwatson.events
ibmicloud.expert
ibm.fi
cieibm-france.fr
ibm.fr
ibmwatson.guru
ibm.hu
nic.ibm
ibm.ie
ibm.co.il
ibm.net.il
ibm.co.in
ibmpackers.co.in
ibmwatson.co.in
ibm.in
ibmglobal.in
ibm.inc
ibm.info
ibm100.info
ibmcenter.info
ibmcloud.info
ibmcorporation.info
ibmsmartbusinesscloud.info
ibmsmartcloud.info
ibmwatson.info
isibm.info
ibm.irish
ibm.it
ibm.com.jm

ibm.co.jp
ibm.jp
ibm.london
ibm.lv
ibm.ma
ibmwatson.marketing
ibm.me
ibm.mobi
ibmuc2.mobi
ibmucc.mobi
ibmucsquared.mobi
ibm.com.mx
ibm.mx
www.ibm.cs186.akadns.net
api.ibm.com.cs186.net
ts-api.ibm.com.cs186.net
ts-api-cdt.ibm.com.cs186.net
ts-api-pre.ibm.com.cs186.net
ibm.net
ibm-smart-cloud.net
ibm-smartcloud.net
ibm-software.net
ibm-watson-for-engineers.net
ibm370.net
ibmcorporation.net
ibmdw.net
ibmexpert.net
ibmexperts.net
ibmfacts.net
ibmlotusprotector.net
ibmserver.net
abn.ibmserver.net
boncurves.ibmserver.net
cosmeticss.ibmserver.net
featherweightpoppy.ibmserver.net
newwaveseawall.ibmserver.net
shop.ibmserver.net
www.shop.ibmserver.net
ibmsmartbusinesscloud.net
ibmsmartcloud.net
ibmsmartercommerce.net
ibmturkey.net
ibmturkiye.net

ibmuc2.net
ibmucc.net
ibmwasc.net
ibmwatson.net
ibmwatsonhealth.net
shop-ibm.net
teamibm.net
ibm.com.ng
ibm.com.ni
ibm.nl
ibmc.nl
ibmforum.nl
ibm.nu
shopibm.nu
ibm.net.nz
ibm.org.nz
ibm.onl
ibm.ooo
myibm.ooo
mail.citizenibm.org
ibmwatson.partners
ibm.com.pk
ibm.com.pl
ibm.pl
ibm.ro
ibm.co.rs
ibm.ru
ibm-remont.ru
ibm.se
ibmwatson.services
ibm.com.sg
ibm.sg
ibm.si
ibmwatson.site
ibm.sk
ibmwatson.solutions
ibmwork.space
ibmwatson.systems
ibmchina.tech
ibmwatson.today
ibmwatson.top
ibm.com.tr
ibm.tt

ibm.tv
default._bimi.ibm.tv
ibmtv.tv
ibm.com.tw
ibm.co.uk
ssl.ibmcloud.co.uk
ibmconnections.co.uk
ibmcognosanalytics.uk
ibm.us
ibm-smart-cloud.us
ibmpowersystems.us
ibmsmartbusinesscloud.us
ibmwatson.us
ibm.com.uy
ibm.vn
ibmwatson.work
ibm.co.za
ibmverse.co.za

Appendix III. List of nominally "IBM-related" FQDNs hosted in AS19574 ("CSC")

ibm.adult
ibm.africa
ibmwatson.at
ibm.expieda.com.au
ibmbigdata.be
greateribm.biz
ibm-cloud.biz
ibm-storage.biz
ibmcloud.biz
ibmstorage.biz
ibmwatson.ch
hpibm.com.cn
ibmreferrals.cn
ibmverse.cn
ibmcognitivebusiness.co
ibmcognitivecommerce.co
ibmcognitiveenterprise.co
99ibm.com
acalabrutinibmaleate.com
xuddpxlbmxibme.adams.com

analyticsibm.com
integration.analyticsibm.com
pyyibmx.aventcorp.com
ibmwebspheremq.axa-im.com
backupibm.com
ibm.bermudatogether.com
betteribmbackup.com
bibmetalbird.com
boulderibm.com
cibmellon.com
cgqhcibmrfeky.crosscountry.com
dolce-ibm-learning-center.com
dolce-ibm-palisades.com
exploreibm.com
fibmiza.com
fuckibm.com
mibmbmo.hexagonmetrology.com
otjwahibmkaypiz.hexagonmetrology.com
hibmenrix.com
ibm-cognitive-business.com
ibm-cognitive-enterprise.com
ibm-cognitive-era.com
ibm-cogntive.com
ibm-corporation.com
ibm-solutions.com
ibm-storage.com
ibm-watson-assistant.com
ibmaccounting.com
ibmartificialintelligence.com
ibmbces.com
ibmbenefitsinfo.com
ibmbigfix.com
ibmbigfixfederal.com
ibmbigfixsmallbills.com
ibmbluemix.com
ibmbrasil.com
ibmbusinessconnect.com
ibmcity.com
ibmclient.com
ibmcognitivebusiness.com
ibmcognitiveenterprise.com
ibmcogntive.com
ibmcoronavirus.com

default._bimi.ibmcoronavirus.com
ibmcovid19.com
ibmdating.com
ibmdoorstraining.com
ibmeast.com
ibmenwillbemen.com
ibmflashsystemcup.com
ibmforthecloud.com
ibmgarage.com
ibmglobalbusinessservices.com
ibmglobalfinancing.com
ibmguardiumfederal.com
ibmhack.com
ibmhospitality.com
ibmibm.com
ibmil-solutions.com
ibminfoprint.com
ibminternet.com
ibminterviewquestion.com
ibmmaelstrom.com
ibmmarketdata.com
ibmmobiledemo.com
ibmmonitoring.com
ibmmydigitalmarketing.com
ibmogilvyserver.com
ibmpalisades.com
ibmq.com
www.ibmq.com
ad.ibmqqdell.com
cb.ad.ibmqqdell.com
pa.ibmqqdell.com
ibmsd.com
ibmsecurityconnect.com
ibmsecuritydemo.com
ibmsecurityfederal.com
ibmsecurityintegration.com
isam.ibmsecurityintegration.com
ibmsidc.com
ibmsmartdata.com
ibmsolutionstore.com
ibmsphere.com
ibmtechnologyconsulting.com
ibmtivoli.com

ibmtranslation.com
ibmuk.com
ibmwatsonassistant.com
ibmwatsonbots.com
ibmwatsonconversation.com
default._bimi.ibmwatsonconversation.com
ibmwatsoncoronavirus.com
ibmwatsoncovid19.com
ibmwatsongroup.com
jos-ibm.com
lenovoibm.com
bibman.onlineikeadesign.com
bmibmr.onlineikeadesign.com
fmpibmgpdikqteune.onlineikeadesign.com
gcaiaiibmcag.onlineikeadesign.com
gribmax.onlineikeadesign.com
ibmexpress.onlineikeadesign.com
ibmonsterdash.onlineikeadesign.com
ibmprod4.onlineikeadesign.com
actribmumb.onlineikeadesigner.com
airajaibmilagros.onlineikeadesigner.com
bibmerge.onlineikeadesigner.com
bibmovil.onlineikeadesigner.com
cribmaster.onlineikeadesigner.com
dotoribms.onlineikeadesigner.com
ebvibmnigst.onlineikeadesigner.com
hussainibmt.onlineikeadesigner.com
iaiibmijffpf.onlineikeadesigner.com
ibm41.onlineikeadesigner.com
ibmctokens.onlineikeadesigner.com
ibmdemo.onlineikeadesigner.com
ibmnvsacc2.onlineikeadesigner.com
ibmpoc1.onlineikeadesigner.com
ibmtjbot.onlineikeadesigner.com
02ibm.openhouseproject.com
allibminnovation.openhouseproject.com
coeibm.openhouseproject.com
ebvibmnigst.openhouseproject.com
ibmagilechampion.openhouseproject.com
ibmail.openhouseproject.com
ibmcicmea.openhouseproject.com
ibmcontract1.openhouseproject.com
ibmengage.openhouseproject.com

ibmlaser.openhouseproject.com
ibmmizuhoprojects.openhouseproject.com
ibmmqa4edlv07.openhouseproject.com
ibmp3.openhouseproject.com
ibmrs.openhouseproject.com
ibmvm.openhouseproject.com
jwepibm.openhouseproject.com
libmiletus.openhouseproject.com
libmms.openhouseproject.com
libmusicxml.openhouseproject.com
novosibmts01.openhouseproject.com
openlibman.openhouseproject.com
237ibm.oregonikeadesign.com
awibmer.oregonikeadesign.com
cb4ibm.oregonikeadesign.com
ibm16v02.oregonikeadesign.com
ibm2.oregonikeadesign.com
ibm4.oregonikeadesign.com
ibmapim.oregonikeadesign.com
ibmfinfwo.oregonikeadesign.com
ibmfs04.oregonikeadesign.com
ibmihgsandbox.oregonikeadesign.com
ibmitproof.oregonikeadesign.com
ibmmeeting.oregonikeadesign.com
ibmnvsprod.oregonikeadesign.com
ibmpawspprd.oregonikeadesign.com
ibmqawards.oregonikeadesign.com
ibmsapis.oregonikeadesign.com
ibmtcp.oregonikeadesign.com
lhslibmediacenter.oregonikeadesign.com
libm1new.oregonikeadesign.com
mibm.oregonikeadesign.com
sjjibm55ya.oregonikeadesign.com
tsuibm0.oregonikeadesign.com
0ibm.oregonikeakitchen.com
aaqibmunir.oregonikeakitchen.com
accountibmdb.oregonikeakitchen.com
akbfwjibm.oregonikeakitchen.com
dlgsasibmico02.oregonikeakitchen.com
gibmail.oregonikeakitchen.com
ibmbzcscjfzrhigw.oregonikeakitchen.com
ibmhcustomers.oregonikeakitchen.com
ibmsoctemp01.oregonikeakitchen.com

iibmedifdang.oregonikeakitchen.com
kmlibm03.oregonikeakitchen.com
libm3.oregonikeakitchen.com
libmanage5.oregonikeakitchen.com
mapslabibm5.oregonikeakitchen.com
novosibmgf06.oregonikeakitchen.com
sibmedia.oregonikeakitchen.com
zemdibmnm.oregonikeakitchen.com
aizqibm.ops.os-light.com
dibmyqvwioe.ops.os-light.com
ibmmiyw.ops.os-light.com
xjbpibm.ops.os-light.com
qbibm.com
rtdqxuvunibmosg.corp.ssv.com
traductionibm.com
ibm.treatchronicanalfissure.com
qmmictibmhwif.ad.urscorp.com
fzpkibmblunbrup.vikingrc.com
jsibmrofvhphl.vikingrc.com
wibmusic.com
txgnaibmgvjrtmf.demabg00.da.de
ysbulgqibma.demabg00.da.de
btglibmdwsleq.orenstein-koppel.de
weibmobile.de
ibm.design
default._bimi.ibm.design
ibmwatson.dev
ibmwatson.dk
ibmbc.es
ibmwatson.es
ibmbigfixsmallbill.eu
ibmbigfixsmallbills.eu
ibmwatson.co.il
ibm-connect.info
ibm-storage.info
ibmbigfixsmallbills.info
ibmcloudcomputing.info
ibmpartner.info
ibmstorage.info
bdlibms.casinoservices.io
ibm.casinoservices.io
ibm-connections.casinoservices.io
ibmdbrxt.casinoservices.io

libm.casinoservices.io
libmassresize.casinoservices.io
libmdesc.casinoservices.io
libmesh.casinoservices.io
libmisc.casinoservices.io
libmongoc.casinoservices.io
libmwaw-zip.casinoservices.io
libmygpo-qt.casinoservices.io
mk-libmad-linux.casinoservices.io
myopencvlibmd.casinoservices.io
pureibm.casinoservices.io
ulibmtp3tests.casinoservices.io
ibmdb.gamingtechnology.io
fuonjmibmishmad.services.lanxess
ibmipqdnyccltb.services.lanxess
yixaiibmajas.services.lanxess
cibmellon.mobi
ibmmobilefirst.com.mx
ibmwatson.com.mx
ibmwatson.mx
acalabrutinibmaleate.net
aibmortgages.net
fibmiza.net
greateribm.net
ibm-watson.net
ibmbigfixsmallbill.net
ibmbigfixsmallbills.net
ibmblu.net
ibmmonitors.net
ibm.instacart-covid19.net
ibm.justeatforbusiness.net
apibmbfjmskllsx.phcgrp.net
ibmhnah.phcgrp.net
saoibm.net
ibmwatson.nu
ibmwatson.co.nz
ibm.online
ibm-cloud.online
ibmstorage.online
ibmwatson.com.pl
ibm.porn
ibm.press
bzz.ibm.press

ibmwatson.ro
acalabrutinibmaleate.ru
ibm-reg.ru
cpanel.ibm-reg.ru
ibmmarkting.ru
ibmservice.ru
ibmwatson.ru
ibmbigfixsmallbills.se
ibm.sex
ibmwatson.com.sg
ibmcoin.top
aibmortgages.co.uk
ibm-servers-storage.co.uk
ibml.co.uk
gjyehezfhwibmzm.proximospirits.us
ibm.website
ibm.xyz
ibmwatson.co.za
ibmxpages.co.za