Realtime DNS Exfiltration Detection in Recursive Resolvers

David Rodriguez [1] Scott Sitar [2] Andrea Kaiser [1] Brian Somers [1] Skyler Hawthorne [1]

[1]Cisco Systems

[2]YeshID

DNS OARC 40

DNS Exfiltration & Tunneling Tools

1. DNSExfiltrator - [1]
2. Iodine - [2]
3. DNSCat2 - [3]
4. dns2tcp - [4]

Let's Exfiltrate

```
$ echo "Hello DNS Oarc 40 and Cisco" \
      | base32 \
      | tr '[:upper:]' '[:lower:]' \
      | perl -pe "chomp;" \
                 "my $n=length;" \
                 "my $i=0;" \
                 "while($i<$n){" \
                    "my $s=substr $_, $i, 7;" \
                    "printf \"@{[($i)/7]}.$s.base32.mydomain.com.\n\";" \
                    "$i+=7;}"
>> 0.jbswy3d.base32.mydomain.com.
>> 1.pebce4u.base32.mydomain.com.
>> 2.zaj5qxe.base32.mydomain.com.
>> 3.yzagqyc.base32.mydomain.com.
>> 4.aylomqq.base32.mydomain.com.
>> 5.eg2ltmn.base32.mydomain.com.
>> 6.xqu===.base32.mydomain.com.
```

DNS Exfiltration & Recursive Resolvers

Enroute a resolver checks the local cache and, if not expired, record served from cache.

Resolver Caches

1. Bind [5] - tree
2. Knot [6] - trie, ranked arrays for rrsets
3. CoreDNS [7] - sharded maps
4. DJBDNS [8] - doubly-linked list, rrsets form linked list

Cache Objectives

Some say read, write others additionally say update, delete

1. Lookup **qnames** from DNS wire format **001a001b003com**
2. Qtypes such as **A, AAAA, TXT, NS, ...**
3. TTLs defining record expiration
4. DNSSEC keys

Cache Lookups

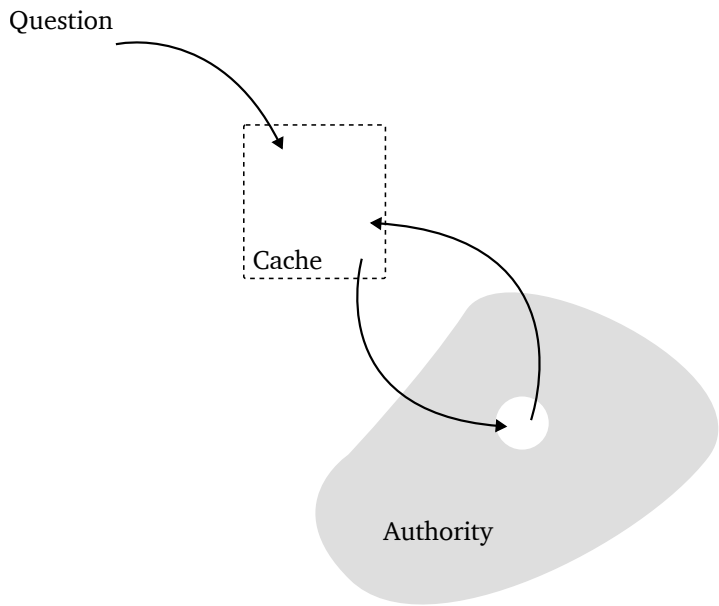

Figure: Cache misses force upstream lookups with answers placed in cache

Realtime Blocklist + Tunneling Cache Lookups

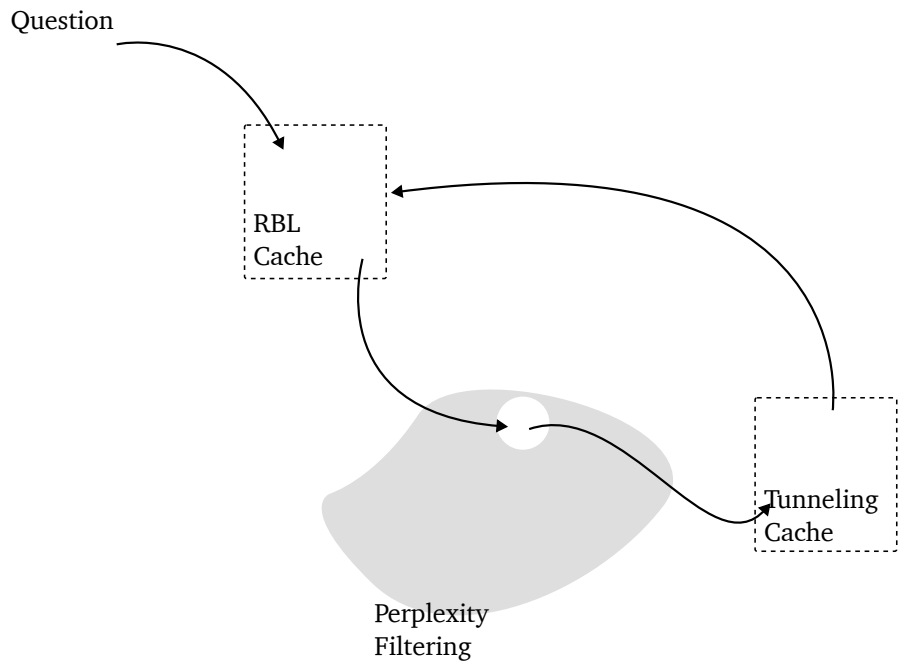

Figure: Cache misses force upstream lookups with answers placed in cache

With two caches we can asynchronously update the RBL cache and isolate performance degradation of the Tunneling cache updates and possibly wait to update until answers are given from upstream name servers.

Perplexity Filter

Given a label walk the string and determine if the word is similar to a known corpus (for example Wikipedia). [9]

$$\frac{1}{n-1}\sum_{i=1+1}^{n-1}[\log(p_{i-1}) - \log(p_{i-1,i})]$$

Example: Let u, b denote a unigram and bigram count function, respectively. Then, given the string xyz we see
$p_0 = u(x), p_1 = u(y)$ and $p_{0,1} = b(x, y), p_{1,2} = b(y, z)$.

Fast Perplexity

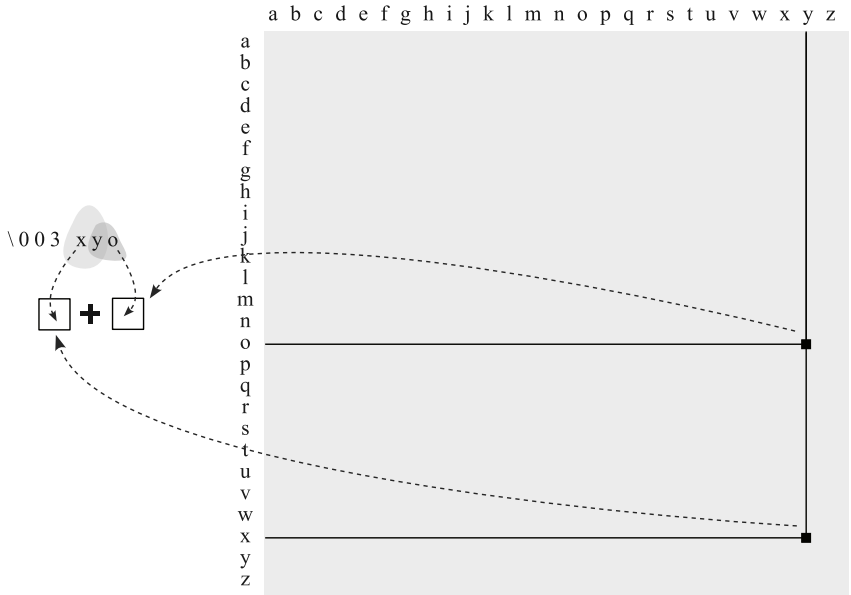

Figure: Perplexity represented as a lookup table requiring only addition and division by $n - 1$

Pseudo-Unique Counts

Let $h : S \rightarrow Z^*$ be a 32-bit murmurhash function, with S the set of all character strings. Then define a fingerprint function as:

$$f(x, n) = 1 << (h(x) \mod n)$$

Explanation: We want a pseudo-set with a fixed size n.
Example:

```
>>> def f(x, n):
...     return 1 << (mmh3.hash(x, signed=False) % n)
...
>>> for elem in list('abc'):
...     print('elem={}, value={}, binary={:b}'.format(elem, f(elem, 10), f(elem, 10)))
...
elem=a, value=1, binary=1
elem=b, value=32, binary=100000
elem=c, value=512, binary=1000000000
```

Detection Algorithm

```
for label IN qname do
    if perplexity(label) > threshold then
        label_fingerprint = fingerprint(label, n)
        key = tunnel_cache_key(qname)
        tunnel_cache_update(key, label_fingerprint)
        break
    end if
end for
```

Operations

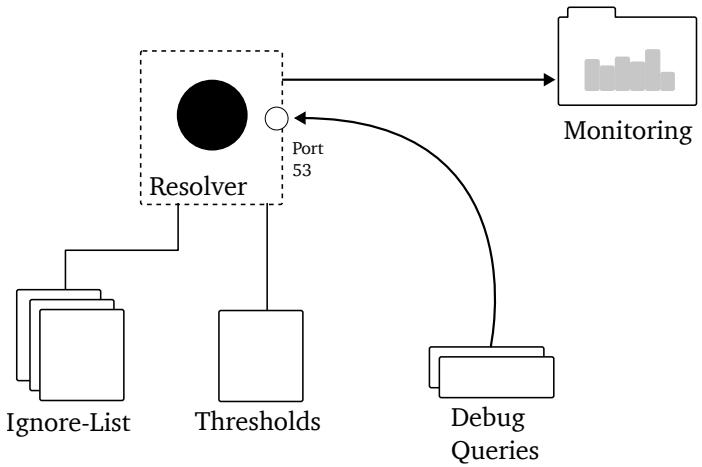

Figure: Operational and tuning of the algorithm occur through allow-lists, thresholds, debugging tools, and monitoring

Challenges: Cache Fragmentation

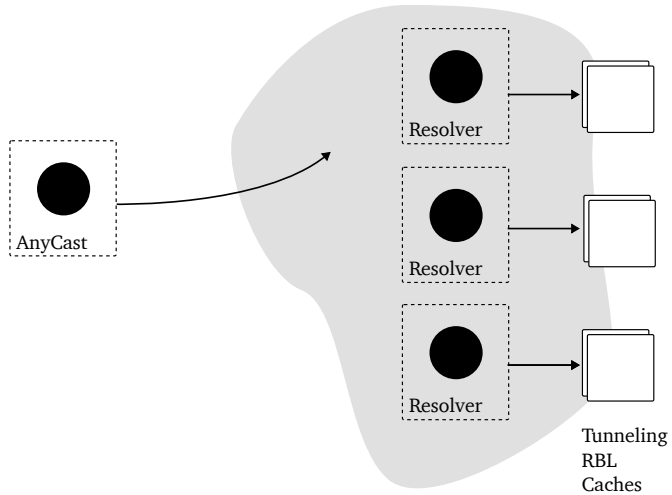

Figure: Aggregating traffic per cache is slower than if all caches shared counter states

Challenges: Cache Key-Pairs

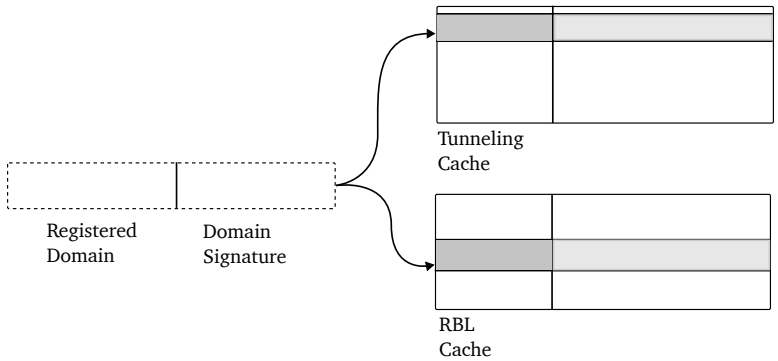

Figure: Cache keys contain some non-obvious information to create unique keys that may eat CPU resources so you don't want to construct multiple times.

Then End!

Enjoy the rest of DNS Oarc

Bibliography

- DNSExfiltrator *https://github.com/Arno0x/DNSExfiltrator*
- Iodine *https://github.com/yarrick/iodine*
- DNSCat2 *https://github.com/iagox86/dnscat2*
- dns2tcp *https://github.com/alex-sector/dns2tcp*
- Bind *https://github.com/isc-projects/bind9*
- Knot *https://github.com/CZ-NIC/knot-resolver*
- CoreDNS *https://github.com/coredns/coredns*
- DJBDNS *https://cr.yp.to/djbdns.html*
- Perplexity and n-gram language models *https://web.stanford.edu/ jurafsky/slp3/3.pdf*