The Collateral Damage of Internet Censorship by DNS Injection

Anonymous [*]
zion.vlab@gmail.com

ABSTRACT

Some ISPs and governments (most notably the Great Firewall of China) use DNS injection to block access to "unwanted" websites. The censorship tools inspect DNS queries near the ISP's boundary routers for sensitive domain keywords and inject forged DNS responses, blocking the users from accessing censored sites, such as `twitter` and `facebook`. Unfortunately this causes collateral damage, affecting communication beyond the censored networks when outside DNS traffic traverses censored links. In this paper, we analyze the causes of the collateral damages and measure the Internet to identify the injecting activities and their effect. We find 39 ASes in China injecting forged DNS replies. Furthermore, 26% of 43,000 measured open resolvers outside China, distributed in 109 countries, may suffer some collateral damage from these forged replies. Different from previous work that considers the collateral damage being limited to queries to root servers (F, I, J) located in China, we find that most collateral damage arises when the paths between resolvers and some TLD name servers transit through ISPs in China.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General

Keywords

DNS, packet injection, Internet measurement, Internet censorship, Great Firewall of China, collateral damage

1. INTRODUCTION

Since DNS is essential for Internet communication, it is a common target for censorship systems. The most popular approach involves packet injection: a censorship system observes DNS requests and injects fake replies to block communication. Yet censorship systems may affect more than just the censored network.

As a concrete example, consider a query for `www.epochtimes.de`[1] from a US user, using a US-based DNS resolver. The US resolver will need to contact one of the DNS TLD authorities for `.de`, located in Germany. If the path to the selected TLD authority passes through China, then the Great Firewall of China (GFC) will see this query and inject a reply which the US resolver will accept, cache, and return to the user, preventing the user from contacting the proper web server.

Packet injection's popularity as a censorship mechanism arises from its ease of implementation. The censor needs to only monitor traffic and inject responses. Thus network operators have used TCP packet injection to block Peer to Peer traffic [6] or undesirable web content [5], and the GFC and others use DNS packet injection to block entire sites. While some ISPs are content to block users inside their network from accessing "unwanted" websites using DNS injection, they may not know that their DNS injecting activities potentially affect users outside their network. In the motivating example of contacting `www.epochtimes.de` from the US, the *collateral damage* was due solely to the DNS request passing through a censored network as traceroute verified that the path for HTTP traffic did not pass through any censored networks.

Although the DNS community has perceived such collateral damage, they only found it happened when resolvers outside contacted DNS authorities inside the censored country [3], with the most famous examples involving queries from Chile that found themselves routed to the Chinese I-root server [8].

However, the range of the potential damage is actually much more complicated. We find that resolvers outside a censored country could suffer from collateral damage caused by DNS injection activities from censored transit networks, even both the resolvers and domains being queried are unrelated to the censored country.

In this paper, we explore the extent of the collateral damage caused by DNS injection. Specifically, we try to answer the following three questions:

(1) How does this collateral damage occur?

(2) Which ISPs are adopting DNS injection?

(3) What names and resolvers are affected?

For the first question, we analyze the causes from the diversity of DNS resolution paths, as well as the dynamic routing. We utilize two tools, *HoneyQueries* to detect affected paths and *TraceQueries* to discover the points of injection. This enables us to identify the censored ASes. Finally, we perform measurements using *StepNXQueries* which allow us to infer whether the collateral damage effect comes from the path between a resolver and the root server or from the path between a resolver and a given TLD server.

A survey of 43,842 non-censored resolvers showed 11,579 suffering from some collateral damage. Unlike the results

[*]The authors wish to remain anonymous and may be reached at the provided email address.

[1]The Epoch Times is a news organization with connections to the Falun Gong spiritual movement. `epochtimes.de` is their German language site.

in [3], we find that the most common source of pollution exists on the paths between the resolvers and the TLD authorities, particularly the paths to `.de` and `.kr` authorities.

The rest of the paper is organized as follows. In § 2 we give a brief introduction to DNS resolution and how packet injection can disrupt the process. Then we analyze the cause for the collateral damage caused by DNS injection in § 3. In § 4 we describe our experiment methodologies and present the experiment results. We have a discussion in § 5 before concluding in § 6.

2. BACKGROUND

The standard DNS resolution process [11, 12, 7] consists of several pieces, including the stub resolver on the user's computer, the recursive resolver, the root servers ("."), Top Level Domain (TLD) authorities, and the site's authority name servers. A typical DNS query process that involves all these servers is illustrated in Figure 1.

If an attacker (e.g. a hacker, an ISP, or a government) has the ability of monitoring any of the steps in the DNS query process, he can inject an additional DNS response(without suppressing the legitimate one), replying with a forged response which has the appropriate query question and ID but with a bogus DNS answer, mapping the queried domain to either an invalid IP address or an IP address controlled by himself. In the absence of DNSSEC validation, the resolver will generally accept the faked answer because it arrives earlier than the real one, and, as a result, the access to the sensitive site will be blocked or redirected.

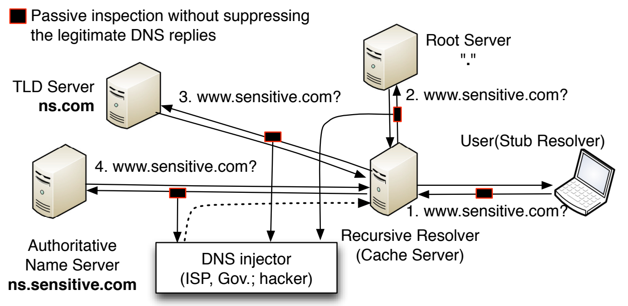
The ease of this attack makes it naturally an effective censorship mechanism. It is well known that the GFC uses this mechanism. A past survey queried more than 800 DNS resolvers in China and found that 99.88% of them were affected by the GFC [9]. And [9] also found that GFC sent tampered DNS responses based on keywords in the domain name. For example, it injects a faked reply for "twitter.computer.com" because "twitter.com" is a blocked domain name.

Unfortunately, the censor appears to over-react to transit DNS queries as well. It inspects all of the transit DNS queries and injects bogus responses, causing collateral damage to non-censored networks. The collateral damage of GFC was first discussed in a DNS operation mailing list when a Chilean operator found that queries from Chile and California to `I.RootServers.NET` sometimes experienced DNS pollution [8]. In [3], Brown et al. analyzed this incident and determined that this kind of pollution could affect many countries because three root DNS server nodes (F, I, and J) have anycast instances in China. They believed that after Netnod withdrew the anycast routes for the Chinese I-root name server from CNNIC, the collateral damage should disappear. However, our work showed this was not the case. We discovered quantities of collateral damage for TLD authorities through dedicated measurement experiments.

3. CAUSES OF COLLATERAL DAMAGE

We assume that DNS censors use over-zealous pattern matching DNS requests, like GFC. Although pattern matching causes a lot of collateral damages(i.e., blocking "twitter.computer.com" because of "twitter.com"), in this paper we focus only on those because of transit DNS queries.

Collateral damage occurs when a DNS query from a recursive resolver enters a censored network, causing the censorship mechanism to react. Although intuition suggests that this would be a rare occurrence, there exists several factors which may cause the censor to receive and react to DNS queries from outsiders.



■ Passive inspection without suppressing the legitimate DNS replies
Root Server "."
TLD Server ns.com
3. www.sensitive.com?
2. www.sensitive.com?
User(Stub Resolver)
4. www.sensitive.com?
1. www.sensitive.com?
Authoritative Name Server ns.sensitive.com
DNS injector (ISP, Gov.; hacker)
Recursive Resolver (Cache Server)

Figure 1: DNS query process and DNS injection

Iterative Queries: A recursive resolver does not send limited queries, such as asking the root for just the name servers of the desired TLD. Instead, if it lacks cache entries for the TLD authorities, it sends query with *entire* domain name to a root server. Similarly, the resolver sends the query with entire domain name to a TLD authority if there are no cache entries for the domain's authority.

This may be further complicated by "out-of-bailiwick" [2] `NS` records. A fairly complicated but not uncommon example is given below. Suppose the DNS authorities for `example.com` are `ns1.example.net` In the absence of cached data, a resolver will handle a query of www.example.com by first querying a root server and later a `.com` TLD authority. The reply from the `.com` TLD will now cause the resolver to query for `ns1.example.net` before resuming the query for `www.example.com`. Thus the resolver will query for `www.example.com` three times: to a root server, to a `.com` TLD server, and to `ns1.example.net`, and at least two queries for `ns1.example.net`: to a root and to a `.net` TLD server. Thus a simple "lookup" may generate numerous queries, and the disruption of any by censorship would cause resolution to fail.

Redundant Servers and Anycast: Most DNS deployments use multiple servers in multiple networks to increase reliability [4], and the actual selection of particular authorities by a recursive resolver is a complex topic, with name servers using various algorithms. Thus, with 13 different roots and 13 servers for the global TLD `.com`, a resolver may experience collateral damage if a path to any one of these 26 IPs passes into a censored network.

Further complicating the picture is the use of *anycast* [13] DNS authorities, where a single IP address may represent a widely deployed system of servers. Two resolvers in different networks may reach different physical servers, along very different paths, even though they are attempting to contact the same IP address.

Censored Transit and Dynamic Routing: The paths from the resolver to the authorities is dynamic, routing through a series of Autonomous Systems (AS). If one transit AS implements censorship, then all traffic which passes through that AS experiences censorship, even if both the source and destination are in non-censored networks. Routing changes also make it difficult to predict when and where DNS queries will pass through censored transit networks.

4. MEASUREMENT AND RESULTS

By measuring of the effect of DNS injection, we want to answer the following two questions related to the collateral damage:

(1) How many ASes implement DNS injection-based censorship?

(2) How widely are DNS resolvers suffering from collateral damage due to censorship, and what is the cause of this collateral damage?

We take three steps to achieve our goals. First, we utilize *HoneyQuery*, scanning the entire Internet to discover injected paths. Then we leverage *TraceQuery* to identify injectors on these paths. Finally we launch *StepNXQuery* which probes open resolvers around the world to evaluate the collateral damage caused by these injectors.

4.1 Searching for Injected Paths

Like the concept of *Honeytoken* [14], we launch a large amount of *HoneyQueries* to search the censored paths. HoneyQueries are DNS resolution requests that include sensitive domain names, and HoneyQueries are expected to receive responses *only* when an AS on the path between the HoneyQuery sender and the destination adopts DNS-based censorship.

Probing Targets: The target of the HoneyQueries are specially selected destinations, which are hosts we know do not run an active DNS server. Thus, normally no DNS responses are expected for these HoneyQueries, and any DNS reply is likely due to censorship related injections.

In order to search all possible AS-level paths, ideally we should make sure that our HoneyQuery probing covers all the ASes in the Internet. We selected an IP address in each /24 of the IPv4 address and verified that the IPs are not running DNS servers. The total list of non-DNS-responsive IPs in our list include 14 million IPv4 addresses, and we call this list the *Probing Targets* of HoneyQuery.

Vantage Point: We selected a virtual private server (VPS) in AS 40676 (Psychz Networks) in US as our vantage point. Previous observers [8, 9] and our own experience indicate that DNS injectors fake answers for both inbound and outbound DNS queries. To further simplify our measurement effort, we assume that an ISP that adopts DNS based censorship technology applies the DNS injection on all the interconnect links between the ISP and all its neighbor networks. With this assumption, our HoneyQuery probing could possibly cover all the ASes from a single vantage point. Although from a single vantage point, HoneyQuery itself can not tell which ASes on a poisoned path made the injection, we use HoneyQueries simply to find enough paths that cover all poisoned ASes. Another tool called TraceQuery (presented in § 4.2) locates the poisoning ASes.

Domain Names used in Probing: Based on our previous experimental study, we select 10 domain names for the probing(Table 1), including social networks, pornography, web hosting, portals, stream media, and search engines, which we would expect to be the targets for government or ISP censorship.

Probing Result: We send HoneyQueries with these sensitive domain names to Probing Targets (14 million non-DNS-responsive IPs) from the vantage point. If there is any valid DNS reply for a HoneyQuery, we mark the domain name as blacklisted and the path to the target IP as a poisoned Path. We also collect all the IPs used as the domain resolution answers in injected responses (we call them lemon IPs). After HoneyQuery probing, we get three lists: (1)`Blacklisted_Domain_List`, containing poisoned domain names in testing domain name set; (2)`Poisoned_Path_List`, containing paths from vantage points to the target IPs suffering directly from censorship; (3)`Lemon_IP_List`, containing the IPs used in all the bogus responses.

Domain Name	Category
`www.google.com`	Search Engines
`www.facebook.com`	Social Networks
`www.twitter.com`	Social Networks
`www.youtube.com`	Streaming Media
`www.yahoo.com`	Portal
`www.appspot.com`	Web Hosting
`www.xxx.com`	Pornography
`www.urltrends.com`	Sites Ranking
`www.live.com`	Portal
`www.wikipedia.org`	Reference

Table 1: Domain Names for Probing.

We conducted our HoneyQuery probing during November, 2011 and obtained a poisoned path list with 388,988 destination IP addresses, distributed in 16 regions (CN, CA, US, HK, IN, AP, KR, JP, TW, DE, PK, AU, SG, ZA, SE, FI) and 197 ASes (We use MaxMind GeoIP database for IP-geolocation mapping). The top regions and ASes are shown in Table 2.

Among these paths which is being poisoned, the large majority (99.80%) paths are with a target IP located in China, which is known to apply DNS based censorship. The rest (about 800) paths are with destinations outside China. It is worth to note that, for each destination IPs of the `Poisoned_Path_List`, it certainly does not mean that their hosting ASes or regions inject faked DNS responses; but means there should be injectors on the paths from our vantage point to them. Our TraceQuery tool actually shows all these 800 paths are results of DNS poisoning from Chinese ASes. § 4.2 describes how we locate the injectors.

We obtained six domain names in the `Blacklisted_Domain_List`: `www.facebook.com, twitter.com, www.youtube.com, www.appspot.com, www.xxx.com, www.urltrends.com`. We also obtained 28 different IPs in the `Lemon_IP_List` [2], by which we can identify the polluted answers from the open resolvers in later measurements(§ 4.3).

4.2 Locating Poisoning ASes

Given the `Poisoned_Path_List`, we identify which AS on the poisoned path injects the faked DNS response with *TraceQuery*. A TraceQuery is a crafted DNS query with a domain name in `Blacklisted_Domain_List` and a customized TTL in its IP header. Like *traceroute*, TraceQuery utilizes TTL decrements to ensure that the packets expire in the network. The query which passes an injector triggers a DNS reply before expiring.

By conducting TraceQuery to the final destination in the `Poisoned_Path_List`, we reveal all the DNS injectors on the path and their locations in the network. Take this path as an

[2] Lemon_IP_List:`http://xixiang.co/lemon-ip-list.html`

AS Number	AS Name	Router IPs
4134	Chinanet	1952
4837	CNCGROUP China169 Backbone	489
4812	China Telecom (Group)	289
9394	CHINA RAILWAY Internet(CRNET)	78
9929	China Netcom Corp.	67
4808	CNCGROUP China169 Beijing Province	55
9808	Guangdong Mobile Communication Co.Ltd.	38
17633	ASN for Shandong Provincial Net of CT	25
4538	China Education and Research Network	22
17816	China Unicom China169 Guangdong province	19
Total 39 ASes		

Table 3: Information of top 10 poisoning ASes.

Region	Count	Percentage
US	12519	28.76
JP	4889	11.23
RU	3306	7.60
DE	2345	5.39
TW	1733	3.98
GB	1580	3.63
CA	1150	2.64
IT	1053	2.42
Total 173 regions		

Table 4: Distribution of open resolvers for StepNXQuery probing.

Region	IP Count	Percentage
CN	388206	99.80
CA	363	0.09
US	127	0.03
HK	111	0.03
IN	94	0.02
Total 16 regions		

(a) Top 5 regions.

AS number	Region	IP Count	Percentage
4134	CN	140232	36.05
4837	CN	88573	22.77
4538	CN	35217	9.05
9394	CN	24880	6.40
4812	CN	14913	3.83
Total 197 ASes			

(b) Top 5 ASes.

Table 2: Statistics of the `Poisoned_Path_List` (identified by IP destinations) collected from HoneyQuery probing.

example: *VantagePoint–AS1–**AS2**–AS3–**AS4**–destIP*, where AS2 and AS4 are deployed with DNS injectors. From the above HoneyQuery, where TTL is set to maximum value by default, we can get two faked replies. To locate all the injectors, we keep sending TraceQueries, increasing TTL one by one, until we get two replies for each query.

After our TraceQuery probing with the 388,988 paths obtained from HoneyQueries (including those 800 paths not addressing to China), we found a list of 3,120 router IPs associated with DNS injection. All the IPs belong to 39 Chinese ASes.

Table 3 shows the information of top ten poisoning ASes. This indicates that all DNS injections are results from China's censorship system.

4.3 Identifying the Injected Query Steps

Given the list of ASes that inject DNS replies, the question remains: to what extent the censorship actions imposed within these ASes affect external resolvers? And during the DNS iterative query process, at which DNS query step does such injection happen?

We probe for such collateral damage using a list of 43,842 open recursive resolvers distributed in 173 countries other than China (Table 4). These open resolvers are collected by probing DNS servers of Alexa Top 1M Websites, combined with open resolvers provided by other researchers.

We probe these resolvers from our non-poisoned vantage-point with domain names derived from the Blacklisted Domain List. Then we compare the IPs in the replies with those in Lemon IP list to see if the resolvers are poisoned. Based on our previous experiments, we found that the DNS censor only poisons UDP-based queries. Thus we probe using TCP, which prevents the censor from poisoning our communication with remote resolvers.

To identify when the resolver triggers injection in the iterative query process, we develop and utilize a series of *StepNXQueries* to detect censored paths between the resolvers and the DNS hierarchy. StepNXQuery takes advantage of over-eager pattern matching in the censorship systems, which regard names such as `www.facebook.com.fu` as objectionable. Through StepNXQuery, we can guarantee that a query from the recursive resolver goes to a specific level in the DNS hierarchy by generating an NXDOMAIN (No Such Domain) triggering request.

To test the root path from the resolver, we query for names like `www.facebook.com.{RANDOM}`, with `RANDOM` being a random string which will trigger a NXDOMAIN response from the root server. By repeating this test 200 times with different random strings, we take advantage of the recursive resolver's willingness to distribute queries among authorities to test as many paths from the given resolver to different roots as possible.

After the root path probing, we find only 1 recursive resolver (124.219.23.209) in AS24154 in TW is poisoned due to the collateral damage.

The same technique allows us to probe the paths between the resolvers and the TLD servers, replacing `{RANDOM}` in above probing with `{RANDOM}.tld`. Before the StepNXQuery probing, we send queries with TLDs to the recursive resolvers, which make the TLD information cached in these resolvers. Thus our StepNXQueries will only traverse the paths between the resolvers and the TLD authorities.

Unlike root probes, the TLDs suffer from substantial collateral damage. We tested all of the 312 TLDs got from ICANN. For the three TLDs in China (`.cn`, `.xn--fiqs8s`, `.xn--fiqz9s`), it is not a surprise that 43,322 (99.53%) resolvers return injected answers.

It is of great concern that 11,573 (26.40%) resolvers showed collateral damage for queries to one or more of 16 other TLDs. Figure 2 shows these TLDs and the number of af-

Rank	Region	Affected Resolvers	Affected Rate
1	IR	157	88.20%
2	MY	163	85.34%
3	KR	198	79.20%
4	HK	403	74.63%
5	TW	1146	66.13%
6	IN	250	60.10%
10	IT	392	37.23%
14	JP	1437	29.39%
16	RU	835	25.26%
18	US	3032	24.22%
20	CA	272	23.65%
25	DE	470	20.04%
Total 109 Affected Regions			

Table 5: In different regions, the open resolvers affected because of querying for blacklisted keywords.

DNS Level	Affected Resolvers	Affected Rate
Root	1	0.002%
TLD	11573	26.40%
Authoritative	99	0.23%

Table 6: Number of affected resolvers in different level.

fected resolvers. The second one, `.xn--3eb707e`, shares the same name infrastructure with the `.kr` ccTLD.

It seems strange that the number of affected resolvers for `.iq`, `.co`, `.travel`, `.no`, `.pl`, `.nz`, `.hk`, `.jp`, `.uk`, `.fi`, `.ca` are all around 90. We check the location of their name servers and find that it is not a coincidence: UltraDNS (AS 12008) hosts some authority servers for all these TLDs except `.hk`.

Limited by space, we only present the detailed information for the most affected TLD: `.de`. As shown in Figure 3, over 70% of the experimental resolvers from KR suffer collateral damage for `.de` queries, such as `www.epochtimes.de`.

Similar to probing TLD servers, we finally constructed queries like `KEYWORD.NXDOMAIN.authority.tld` (e.g., `www.twitter.com.abssdfds.ibm.com`) to explore paths from the resolvers to authoritative name servers for several domains.

We select 82 top popular domains from Alexa sites (outside of China). We see that queries for six domains could potentially trigger censorship on 30–90 resolvers, as shown in Figure 4. Although the numbers of affected domains and resolvers seem small comparing to the results of TLDs testing, this may only represent the tip of the iceberg, considering the over-zealous pattern matching adopt by censorship and the huge number of domain names in the whole Internet.

4.4 Further Analysis on Measurement Results

Table 5 and Table 6 give the total number of resolvers suffering from collateral damage because of paths to root, TLDs and the top 82 domain names. 26.41% of the experimental resolvers are polluted, and they are distributed in 109 regions. The most affected country is Iran, 88.20% of its experimental resolvers suffer the collateral damage.

Unlike the worries presented by Mauricio [8], Table 6 shows that the primary damage arises from censored transit paths to TLD servers. Our result partly confirms Mauricio [8]'s claim that the operator of I-Root server, Netnod, "withdrew their anycasted routes until their host (CNNIC) could secure assurances that the tampering would not recur". Besides, since the roots themselves are highly anycasted, it is unlikely that a path to a root needs to go through China.

To find out why the collateral damage happened, we construct the topology of ASes neighboring CNNIC in Figure 5 using the data from the project of Internet Topology Collection [10]. According to Figure 5, AS31529, which is the AS of a `.de` TLD server (194.0.0.53), is a customer of CNNIC AS24151. Meanwhile, AS24151 is also customers of other foreign ASes. As a result, traffic from foreign ASes to `.de` TLD server may pass through China, and then the collateral damage happens. We illustrate this with the following case. We choose a node from lookinglass [1], which lies in the same AS (AS39737) in Romania as an affected resolver (`89.37.120.6`) does, and review the AS paths to the 6 TLD servers for `.de` from BGP data. Finally, we find that the AS path from AS39737 to a `.de` TLD server (`a.nic.de`,`194.0.0.53`) goes through a censoring AS (AS7497) in China, which is the cause of the collateral damage on this resolver. We show the AS path in Figure 5: 39737, 6939, 10026, 7497, 24151, 31529.

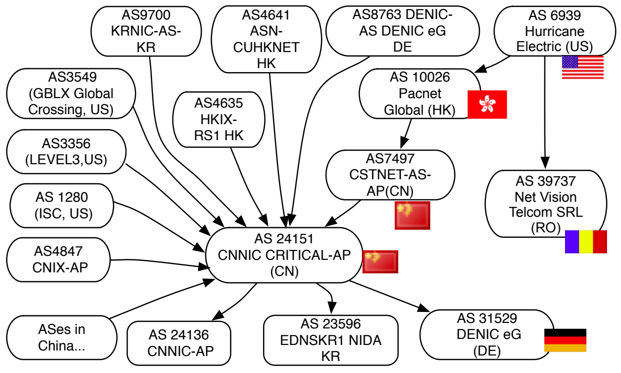


Figure 5: Topology of ASes neighboring CNNIC

5. DISCUSSION

The cause of the collateral damage presented in this paper is the censorship activities by ISPs providing transit, not just connectivity. We hope that this paper will raise the awareness of the collateral damage caused by indiscriminate DNS censorship.

To avoid the collateral damage while keeping the censorship policies, one possibility would be for the ISPs to apply more strict checks to avoid polluting transit queries. If ISPs only censor the customers, not the transit, they may prevent the collateral damage. However, because of the closed nature of many censorship activities (such as the DNS filter in China), it is unclear to us if there are any technical challenges for those ISPs to implement such policy or not.

If the censoring ISPs do not change their current practice of DNS-injection, another possibility is for neighboring ISPs to consider them invalid for transit: the neighbors should prefer alternate paths and not advertise transit whenever an alternate path exists. In particular, the TLD operators should monitor their peering arrangements to check for censored paths.

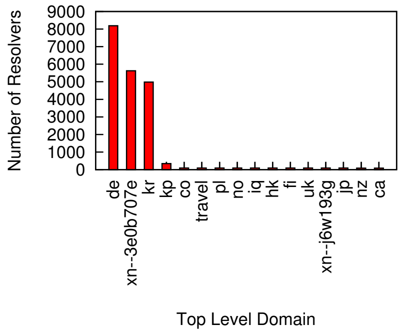


Figure 2: Number of affected resolvers for TLDs.

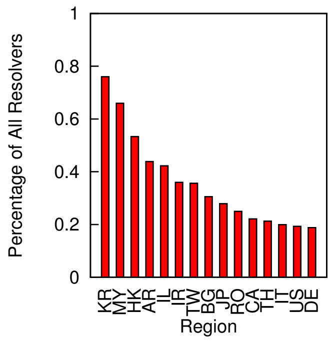


Figure 3: Distribution of affected resolvers for TLD `.de`.

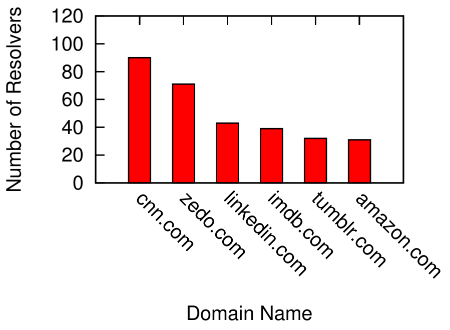


Figure 4: Number of Affected resolver for authoritative domains.

Finally, and most importantly, DNSSEC naturally prevents this collateral damage, especially on the TLD level. Both the `.de` and `.kr` domains sign their results, thus enabling a DNSSEC-validating resolver which rejects the unsigned injected replies while waiting for the legitimate signed replies could avoid suffering collateral damage due to packet injection.

6. CONCLUSION

The contributions of this paper include:

(1) Detailed analysis of collateral damage caused by DNS injection. Iterative queries to different levels of name servers, multiple name servers distributed in different locations and dynamic and anycast routing, are all factors which may cause a query to transit a censored network, even though both the user and the target are outside the censored area.

(2) Discovering and locating DNS injectors. We probed all the Internet to find the indiscriminate DNS injectors, locating these DNS injectors in 39 Chinese ASes.

(3) Measurement of affected recursive resolvers all over the world. We measured 43,842 open recursive resolvers in 173 countries, and found 26.41% of them in 109 countries could be polluted for some blacklisted keywords.

(4) Primary path of pollution: from resolvers to TLD servers. We find that the primary collateral damage arises from transit between the resolvers and the TLD authorities, particularly the authorities for `.de` and `.kr`.

We expect to continue our study on the measurement of the collateral damage caused by DNS injection, using multiple vantage points and an expanded list of HoneyQueries. And if GFC changes its DNS injection policy(such as moving to suppress other legitimate responses) later, we have to use a different methodology to adapt to the change. Although we has not come to a solution to allow recursive resolvers to be immune to the collateral damages from DNS-based censorship apart from DNSSEC validation, we hope our result can increase the Internet communities' awareness of such behaviors and urge them to take actions to actively detect and resist such pollution to the whole Internet.

7. REFERENCES

[1] LookinGlass. `http://lookinglass.org`.

[2] Technical requirements for authoritative name servers. `http://http://www.iana.org/procedures/nameserver-requirements.html`.

[3] M. A. Brown, D. Madory, A. Popescu, and E. Zmijewski. DNS Tampering and Root Servers, Nov. 2010. `http://www.renesys.com/tech/presentations/pdf/DNS-Tampering-and-Root-Servers.pdf`.

[4] R. Bush, M. Patton, R. Elz, and S. Bradner. Selection and Operation of Secondary DNS Servers. *RFC2182*, 1997.

[5] J. Crandall, D. Zinn, M. Byrd, E. Barr, and R. East. ConceptDoppler: A Weather Tracker for Internet Censorship. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS'07, pages 352–365, New York, NY, USA, 2007. ACM.

[6] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 27–27, Berkeley, CA, USA, 2010. USENIX Association.

[7] R. Elz and R. Bush. Clarifications to the DNS Specification. *RFC2181*, 1997.

[8] M. V. Ereche. Odd Behaviour on One Node in I root-server, 2010. `https://lists.dns-oarc.net/pipermail/dns-operations/2010-March/005260.html`.

[9] G. Lowe, P. Winters, and M. L. Marcus. The Great DNS Wall of China. `http://cs.nyu.edu/~pcw216/work/nds/final.pdf`, 2007.

[10] W. Lucas. Internet Topology Collection. `http://irl.cs.ucla.edu/topology/`.

[11] P. Mockapetris. Domain Names - Concepts and Facilities. *RFC1034*, 1987.

[12] P. Mockapetris. Domain Names - Implementation and Specification. *RFC1035*, 1987.

[13] C. Partridge, T. Mendez, and W. Milliken. Host Anycasting Service. *RFC1546*, 1993.

[14] L. Spitzner. Honeytokens: The Other Honeypot. `http://www.symantec.com/connect/articles/honeytokens-other-honeypot`, 2003.