Project Report

DNS Injection: Detection and Analysis

CSE 524 : Advance Project II

Under Guidance of:
Prof Phillipa Gill

By
Arpit Singh
(110162005)

Department of Computer Science
Stony Brook University

Table Of Contents

1. Introduction:

DNS injection is a popular censorship mechanism deployed by many governments to block certain websites. However such blocking often affects the traffic that passes through censored network.
In DNS injection, DNS requests are monitored and fake replies are injected in case the requested domain name is one of the blocked domains. However it is interesting to note that not just the site is blocked if , we send request of domain which contains blocked domain as substring that too get blocked. Thus , domain names which by coincidence contains blocked domain as substring will be subsequently blocked. In this project, i have developed tools and techniques to measure DNS injection.I have run experiments in my limited setup to verify the results. The motivation of work is taken from anonymous paper based on the collateral damage of internet censorship by DNS injection[1]. I have verified my results by running against various IP address as well as focusing on china based IP address. Step by step method to analyze the DNS injection is described as part of work. Ideas are also taken from another paper based on DNS injection[2]

2. Study of DNS injection:

To study DNS injection, we start with keeping three question in mind:

a. How to detect DNS injection?
b. How to detect Routers and AS which are causing DNS injection?
c. What is the extent, level and effect of DNS injection?

In order to answer first question, program called "Honey Queries is developed". In honey query we send DNS requests to IP address which are in high probability not running DNS servers.If DNS injection is not present , we will not receive a reply. However if we gets reply that means DNS injection is present in network and one of routers of DNS injecting network is replying. All IP address ranges allocated in target range are obtained and all possible /16 address are generated. Each address in targeted network is pinged to find out if DNS injection is present or not.

For, second question, we developed Trace queries. Based on the result of Honey queries, we will have result of IP address and domain name for which we see injected response. Now we use this data and start sending DNS request to given IP address with increasing TTL.We record the TTL at which we get reply. TTL indicates how many hopes are needed to get injected reponse indirectly indicating hops required to reach IP address which might have injected the poisonous response. We run traceroute on IP address and based on TTL value, get the IP address of router which might have sent out injected response.

For third questions, we developed Step NX queries, in which we create multiple forms of blocked domain(prefixed,suffixed) and send DNS queries to IP address obtained from honey query to determine whether Substrings are blocked too. Also Honey queries are send to DNS resolvers across the world with domain name containing substring as blocked IP to determine collateral damage.

3. Honey Queries:

Honey queries is term for DNS request which tends to send DNS request to IP address which are not DNS servers. If reply comes back, it is indicator of DNS injection.
Following steps are taken to implement honey queries.

a. Collection of IP address by country.

As first step, we need IP block of all countries so that we can target individual country, individual block as well as full country itself. The list of blocks of IP address is obtained from website http://www.nirsoft.net/countryip/ .
The website lists all ip address blocks country wise and also list the AS who owns them. IP address blocks of all countries is collected and stored as CSV files.

b. Generation of IP address in /8 and /16 range :

After collecting IP address, next step is to generate all IP address in /8 or /16 range depending upon the rate of queries. Following script is used for IP address generation:

Input is of form: 125.110.0.0,125.110.255.255,65636 (start address,end address and no of address in range).
It is the csv file obtained directly from above mentioned website

```
//Script for generating IP address in /16 blocks:
import math
import csv
from netaddr import *
import pprint
import math
import socket,struct
import sys
input_file = str(sys.argv[1])

file_open = open(input_file,'r')

bin_8 = '10000'

filename = "output"

output = open(filename,'w')

with open(input_file,'rb') as f:
        for row in csv.reader(f,delimiter=',',skipinitialspace=True):
                temp = row
                if temp[0] == "" or temp[0].replace(" ","") == "":
                        continue
                add_start = row[0]
                add_end = row[1]
                no_of_add = int(row[2])


                ip_add = IPAddress(add_start)
                no_of_gen_add = no_of_add/16

                for i in range(0,no_of_gen_add):

                        writen = add_start + '\n'
                        output.write(writen)

                        add_bin = ip_add.bin

                        bin_add = int(add_bin,2)+int(bin_8,2)

                        add_start = socket.inet_ntoa(struct.pack('!L', bin_add))
                        ip_add = IPAddress(add_start)

//Scripts Ends
```

However, For China alone, no of generated address :4,15,43,680.
For iran, number of generated address : 13,45,024.
Since it was not possible to run script for all address , i focused on specific AS to find out DNS injection.
I focused on Chinanet and generated 4102 address to send DNS requests in address range (125.110.0.0,125.111.0.80)

Thus i have list of IP address to find evidence of DNS injection.

c. DNS-Python :

Next step is to write scripts for DNS injection itself. I initially used DNS-python(www.dnspython.org). It was initially created by company named Nominum and later made open source
DNSpython is library to support DNS queries. It supports all record types used for queries,zone transfer and dynamic updates. It can be used for high level as well as low level access.The advantage of using DNS-Python is that it is one of its kind DNS specific library and all nitty gritty of DNS features can be varied to craft DNS packet . Also it does not use RAW sockets, thus allowing me to work on machines on which i do not have root access. However disadvantage is we cannot modify IP level parameters like TTL. Initially to overcome this hurdle i even tried to vary TTL of the whole system in order to send out queries of varying TTL.

d. Scapy:

Scapy(http://www.secdev.org/projects/scapy/doc/usage.html) is another library that can be used to generate DNS requests. Scapy can be used to generate any type of request as it is packet crafting library.It is powerful packet manipulation library with help of which we can forge/decode various types of packet. However it creates RAW socket , thus we need root access to run scapy.

e. Script for sending DNS request to specified IP address:

I used DNS python to send request to specified list of domains and IP address. Following is the script i used to query IP address obtained in earlier steps.:

The input to script is list of domain names and list of IP address generated above

```
//Script to send DNS request to list of IP address
import dns.resolver
import os
import sys

d = open(str(sys.argv[1]),'r')
ip = open(str(sys.argv[2]),'r')
domain_list = d.read().splitlines()
ip_list = ip.read().splitlines()
my_resolver = dns.resolver.Resolver()
output_f = open("output.txt",'a')
for nameserv in ip_list:
        nameserv = nameserv.replace("\n","")
        my_resolver.nameservers = [nameserv]
        for domain in domain_list:
                try:
                        domain = domain.replace("\n","")
                        answer = my_resolver.query(domain).response
                        #print answer.response.answer
                         entry = "Querry start \n" + "server : " + "    " + nameserv + " " +
"query_result " + str(answer) + "\nQuerry ends "+ '\n \n'
                        output_f.write(entry)
                except :
                        print domain,nameserv
                        print "no domain"



d.close()
ip.close()
output_f.close()

//Scripts ends here
```

Following domain names are used for DNS injection:

www.facebook.com
www.twitter.com
www.youtube.com
www.appspot.com
www.xxx.com

www.urltrends.com.

The list of ip address is list of chinanet address obtained in previous results

f. Collection of results:

Above scripts provides result in format:

```
//Result format
Querry start
server :    125.110.0.240 query_result id 40885
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
www.twitter.com. IN A
;ANSWER
www.twitter.com. 1856 IN A 159.106.121.75
;AUTHORITY
;ADDITIONAL
Querry ends
//Result format ends
```

This format is chosen since all output can be recorded and used for future purpose.
Above output is formatted to get desired format of dns-server, domain -name and ip-address obtained using following script:

```
//Script to parse results in desired format:
import sys
import os


filename = str(sys.argv[1])
out = open(filename,'r')
l = out.readlines()
honeyquery = open("honeyQuerryOutput",'w')
i = 0
while i < len(l):
        line = l[i].split()
        if "start" in line:
            i += 1
            line = l[i].split()
            server = line[2]
            i = i+5
            line = l[i].split()
            domain = line[0].replace("com.","com")
            lemonlist = list()
            #print server,domain,line

            i += 2
            line = l[i].split()
            if "IN" in line and "A" in line:
                lemonlist.append(line[len(line)-1])
                i = i+ 1
                line = l[i].split()
            else :
                i = i+1
                line = l[i].split()

            while "IN" in line and "A" in line:
                lemonlist.append(line[len(line)-1])
                i = i+ 1
                line = l[i].split()

            for item in lemonlist:
                entry = server+','+domain+','+item+'\n'
                honeyquery.write(entry)

        i += 1
        if (i > len(l) -10):
```

//Scripts ends which parses result in desired format.

Above script produces result in following format:
(Nameserver,domain name, lemon IP)
125.110.0.192,www.facebook.com, 159.106.121.75

g. Analytics of Results:

Thus on running above script we obtained 682 results.
Interestingly all websites are blocked almost equally .In one instance, we obtain true IP address of urltrends.com. Also we obtained 10 different Lemon IPs

(I) Lemon IPs. Following table summarizes occurrence of Lemon IPs.

LemonIP	No of Occurrence	Location of IP
203.98.7.65	26	Vodafone New Zealand
159.106.121.75	505	DoD Network Information Center,USA
37.61.54.158	20	Baktelekom, Azerbaijan
78.16.49.15	24	BT Communications Ireland
103.224.182.16	1	Actual host server of urltrends
243.185.187.39	23	invalid ip address
46.82.174.68	22	Deutsche Telekom AG Germany
8.7.198.45	20	Level 3 Communications North America
59.24.3.173	23	Korea Telecom Korea
93.46.8.89	18	Fastweb Europe

Thus we see 9 invalid IP address returned. One of address which corresponds to department of defence IP address is used heavily. While other IP address are returned randomly and used almost equally.

All IP address correspond to various countries. However usage of one particular IP address which happens to be US DoD IP address is intentional or not, is not clear.
Other IP address correspond to New Zealand, Ireland,Korea etc. and sent in random manner.
The website used to locate IP address is : http://whatismyipaddress.com/ip-lookup

(II) Websites Blocked

Domain	No of occurrence
www.facebook.com	138
www.urltrends.com	1
www.youtube.com	137
www.twitter.com	135
www.appspot.com	135
www.xxx.com	136

Thus all suspected websites are blocked by chinanet .It looks urltrends is not blocked as its one instance correspond to true IP address, which seems to indicate one of the request hit actual DNS server. Since other websites are blocked we see injected responses.

4. Trace Queries:

Trace query are DNS request sent to IP address obtained from honey queries with increasing TTL. TTL can later be used to determine router IP sending injected responses.

a. Determining TTL for pinged IP:

Script is written to determine TTL. We have used Scapy to determine to generate packets with varying TTL and send out address obtained from honey query output.

Following is the script to determine TTL of queried response:
Input to the script is in form of honey query.
It is in following format:
(dns-queries, domain-name,returned address)

```
//Scripts to determine TTL
import sys
from scapy.all import sr1,IP,UDP,DNS,DNSQR
import os
import sys

honeyout = open(str(sys.argv[1]),'r')

trace_ttl_out = open("trace_ttl_amazon.txt",'a')

ttl_val = 2

honey_list = honeyout.read().splitlines()

for entry in honey_list:
        entry_list = entry.split(",")
        nameserv = entry_list[0]
        qry = entry_list[1]
        print nameserv
        print qry

        for ttl_val in range(14,30):

p=sr1(IP(dst=nameserv,ttl=ttl_val)/UDP()/DNS(rd=1,qd=DNSQR(qname=qry)),retry
=0,timeout=2)
                try:
                        if p[DNS].ancount >= 1:
                                entry_t = nameserv+','+str(ttl_val)+','+qry+'\n'
                                print entry_t
                                trace_ttl_out.write(entry_t)
                                break
                else :
                        print "No data found"
        except:
                continue
```

//Scripts ends

Output of query is in following format:
(dns-queries,TTL-value,domain-name)
125.110.29.0,23,www.youtube.com

In total we have obtained 423 results from above input. Following is sample output:

```
125.110.29.16,27,www.facebook.com
125.110.29.16,19,www.xxx.com
125.110.29.32,19,www.facebook.com
125.110.29.32,16,www.twitter.com
125.110.29.32,18,www.youtube.com
125.110.29.32,19,www.appspot.com
125.110.29.32,18,www.xxx.com
125.110.29.48,17,www.facebook.com
125.110.29.48,17,www.twitter.com
125.110.29.48,17,www.youtube.com
125.110.29.48,14,www.appspot.com
125.110.29.48,17,www.xxx.com
125.110.29.64,19,www.facebook.com
125.110.29.64,19,www.twitter.com
125.110.29.64,19,www.youtube.com
125.110.29.64,19,www.appspot.com
```

We see different TTL for same dns server indicating multiple layers of routers injecting DNS

b. Running Traceroute on pinged IP .

In order to obtain IP address of router involved we used traceroute python script to obtain ip address of routers (https://github.com/ayeowch/traceroute) Since TTL obtained often encompsses total IP address available. If no of ip address available is less that TTL, then last hop is considered as source of injected DNS response.

c. Finding IP address of poison routers.

Thus individual IP address and all details of dns injecting router can be obtained using following script:

```
\\script for finding traceroute IP address
import os
import sys
from traceroute import Traceroute
tracefile = open(str(sys.argv[1]),'r')
traceline = tracefile.readlines()
router_list = []
trace_out = open("trace_output.txt",'a')
for line in traceline:
        entry = line.split(',')
        d_name = str(entry[0])
        ttl_val = int(entry[1].rstrip())
        traceroute = Traceroute(d_name)
        try :
                hops = traceroute.traceroute()
                total_hops = len(hops)
                if total_hops < ttl_val :
                        out_hop = hops[total_hops-1]
                        router_list.append(hops[total_hops-1])
                else :
                        router_list.append(hops[ttl_val-1])
                        out_hop = hops[ttl_val-1]

                output = out_hop['ip_address'] + '\n'
                trace_out.write(output)
        except:
                continue
print router_list
//Script ends here.
```

The script gives output and details of all IP in following format:

```
{'hostname': '38.88.197.6', 'longitude': -118.4041, 'rtt': '113.101 ms', 'hop_num': 10, 'latitude': 33.9571, 'ip_address': '38.88.197.6'}
```

However due to some bug in original script itself, it does not work for IP address ending with 0.

The list of router ip address is also obtained in separate file.

Hence we have obtained list of all router IP address causing DNS injection.

In my current setup, i have obtained total of 459 routers IP. Among them i have 46 unique entries .Many of IP address are part of china Backbone network. Often traceroute could not access all address in path via ICMP. Hence few address in list does not seem right due to inability of traceroute to obtain all address

Few of the prominent entries are given below in table:

Router IP injecting response	No of instances
154.54.0.237	53
154.54.88.10	36
202.97.49.105	32
202.97.50.21	28
202.97.50.1	25
202.97.50.25	17
54.54.5.66	16
38.88.197.6	15
202.97.50.29	15
202.97.90.113	15
38.88.197.86	14
38.88.197.10	14
38.88.197.58	13
38.104.83.198	13
38.104.210.186	12
154.54.7.54	11
38.122.146.146	11
202.97.58.177	9
38.122.146.142	9

5. Stepnx queries:

Step nx queries are used to determine impact of blocked IP address. Thus we generate combinations of domain names and send them to well known domain names. It is interesting to get responses of domain names which does not exist.

a. Generation of Combination of domain names and sending DNS queries to IP address :

As first step of stepnx queries , we wrote a script to generate random combination of IP address .

Following script is used to generate 5 random combinations of domain names by prefixing as well as suffixing domain names.

//Script generating combination of domain names and sending DNS requests:

```
import dns.resolver
import os
import sys
import random
import string

d = open(str(sys.argv[1]),'r')
ip = open(str(sys.argv[2]),'r')
domain_list = d.read().splitlines()
ip_list = ip.read().splitlines()
my_resolver = dns.resolver.Resolver()
output_f = open("output.txt",'a')
for nameserv in ip_list:
        nameserv = nameserv.replace("\n","")
        my_resolver.nameservers = [nameserv]
        for domain in domain_list:
                #try:
                        #domain = domain.replace("\n","")
                        domain_append = list()
                        domain_parts = domain.split(".")
                        if len(domain_parts) < 3:
```

```
                continue
        dom = ''.join(random.choice(string.ascii_lowercase) for i in range(5))
        str1 = domain
        str2 = domain_parts[1] + '.' +domain_parts[2]
        str3 = dom+str2
        str4 = dom +'.'+str2
        str5 = str2+'.'+dom
        domain_append.append(str1)
        domain_append.append(str2)
        domain_append.append(str3)
        domain_append.append(str4)
        domain_append.append(str5)
        print domain
        print domain_append
        for item in domain_append:
                try:
                        answer = my_resolver.query(item).response
                        #print answer.response.answer
                         entry = "Querry start \n" + "server : " + "   " + nameserv + "
" + "query_result " + str(answer) + "\nQuerry ends "+ '\n \n'
                        output_f.write(entry)
                except :
                        print item,nameserv
                        print "no domain"
                        continue

//Scripts ends
```

b. Collection of target IP(DNS Servers):

The list of target IP address consists of well known DNS servers. List of well known DNS servers is obtained from (http://public-dns.tk/) which contains public dns server list.
I tried contacting (http://openresolverproject.org/) to get the list of open resolvers. However, several reminders and mails are not responded. Hence i decided to use DNS servers and see if injection is present.

c. Result and Discussion.

The result of above query is parsed in same way as we have parsed the honey query
On querying various DNS servers from china, i obtained following results:

```
183.196.3.25,ppmflyoutube.com,159.106.121.75
183.196.3.25,ppmfl.youtube.com,159.106.121.75
183.196.3.25,www.appspot.com,159.106.121.75
183.196.3.25,appspot.com,159.106.121.75
183.196.3.25,spptpappspot.com,159.106.121.75
183.196.3.25,spptp.appspot.com,78.16.49.15
183.196.3.25,www.xxx.com,159.106.121.75
183.196.3.25,xxx.com,159.106.121.75
183.196.3.25,lasco.xxx.com,159.106.121.75
183.196.3.25,www.urltrends.com,103.224.182.16
183.196.3.25,urltrends.com,103.224.182.16
183.196.3.25,vkcmc.urltrends.com,103.224.182.16
 58.132.8.1,www.facebook.com,59.24.3.173
 58.132.8.1,facebook.com,37.61.54.158
 58.132.8.1,hlwiu.facebook.com,37.61.54.158
 58.132.8.1,www.twitter.com,37.61.54.158
 58.132.8.1,twitter.com,243.185.187.39
```

Thus we see, Injected DNS responses when domain name contains just substring of full domain names.
Thus it can be concluded, censorship not only applies to blocked domain names but also the domain names which contain the censored name as substring. Above two ip address 183.196.3.25 are well known DNS server and it is highly possible in between injected DNS response is received rather than reply from original DNS server.
For every domain names, we have tried 5 combinations of domain names, which includes prefix addition,suffix addition, original IP, prefix after dot and suffix after dot.We see injected response in every query indicating a collateral damage of domain names which contains blocked domain names as substring in any form.

6. Experimental Setup:

All experiments are conducted on standard ubuntu machines with 8GB RAM running on Intel® Core™ i7-4510U CPU @ 2.00GHz × 4 running version 14.04 running Linux kernel 3.19.
Apart from Local machine, Google Cloud machine and Amazon EC2 cloud machine is also used to run experiments. Both the cloud machines has 8GB RAM.

7. Conclusion :

Thus we have designed the setup,toolkit for detecting DNS injection. Experiments were run to detect injection and its collateral damage. We ran honey querry to detect widespread DNS injection in chinese network.Next we run Trace queries to detemine TTL at which we get DNS response. TTL value can be subsequently used to determine router IP resulting in injected response.
Step NX queries ran to determine collateral damage on domain names which contain the censored ip address as substring.

8. Future Work:

As part of future work, rather than running honey query on al IP address, it is suggested to run honey queries on set of IP address of particular country with particular ISP. After wasting many hours on running honey query on unused ip address, i realized best option is to run on set of selected suspected IP address in order to reach results fast. It works in way that we need to first select countries where blocking is suspected then narrow down in state owned ISP. Required ip address in \16 range can be generated and subsequently honey queries can be sent out. Further analysis can be done based on results of honey queries. Possible countries that need to be tested include Iran,Saudi Arab ,middle eastern countries and north African countries.

9. References

1. Anonymous. (2012). The collateral damage of internet censorship by DNS injection. SIGCOMM Comput. Commun. Rev. [Online]. 42(3), pp. 21–27. Available: http://doi.acm.org/10.1145/2317307.2317311
2. Matth¨aus Wander, Christopher Boelmann, Lorenz Schwittmann, and Torben Weis. Measurement of Globally Visible DNS Injection. Access, IEEE, 2:526–536, 2014