Network Administration

Short Answers

1. The reasons may be a load balancing feature, or the redundancy, of the chosen hosting provider. To serve web pages based on the user location (asia, europe, etc.) might be the reason, too.

References

[1] Why do I see two different IP addresses for a website?

[2] Why does a domain name identify one or more IP addresses?

2. When is cache used: after the resolver sends a query to local DNS, it takes many queries before local DNS sends back the answer. Taking performance into accounts, DNS server will caching results locally, with the time to live (TTL), an expiration time after which the results must be discarded or refreshed. As mentioned above, recursive DNS servers are using cache to save time of queries. In the other hand, authoritative DNS server also needs caching to help dealing with millions of queries at the same time.

References

[1] The introduction of DNS

[2] DNS Caching and How It Makes Your Internet Better

[3] Authoritative DNS Servers vs. Recursive DNS Servers

[4] The operation of DNS

3. When making a DNS change, it takes time for the changes to take effect. This is called DNS propagation—the time it takes for the domain DNS to refresh the cache on the network. The cache is cleared over a certain amount of time called ‘time to live’ (TTL). When the DNS refreshes according to its TTL, the propagation is complete.

   - High TTL: We can set up longer TTL to mitigate the workload of the DNS server. High TTL means the DNS servers will answer queries in lower frequency. Longer TTL’s also cut resolution times. Every time a query has to ask an authoritative name server, it adds an additional lookup, which could add precious milliseconds. Records that point to the web server or CDN, A and CNAME records should have longer TTL since they’re rarely changed. Also, some service providers charge on a usage basis for DNS lookups.
   - Low TTL: The sooner the cached record expires. This allows queries for the records to occur more frequently. If we’re going to make DNS changes, we should set up TTL as low as possible. Additionally, lowering the TTL is a good idea for using DNS for failover.

References

[1] Propagation - How Long do DNS Changes Take?

[2] What are the benefits of a high TTL for DNS?

[3] Best practices for using TTL

[4] zhTW-Wiki Failover

[5] TTL Best Practices: the Long and Short of It

4. Threat Model: The attacker may invade the DNS server to manipulate DNS resolver's cache, causing the name server to return an incorrect result record, e.g. an IP address. This results in traffic being directed toward the attacker's or any other computer. The attacker can fake a website that looks alike the authentic website to spoof the user's account and password.
   DNSSEC use hash function and public-key cryptography to achieve the following three features:

   - Data Integrity: When DNSSEC is used, each answer to a DNS lookup contains an RRSIG DNS record that is a digital signature of the answer DNS resource record set. The receiver is able to use public key and RRSIG to verify whether it is the original RR record and is not modified.
   - Origin Authentication of DNS Data: This is a question about whether the DNS server is malicious or not. The DNS server must send its digital signatured public key (DNSKEY) to its parent zone server; this new RR type is called Delegation Signer(DS). We can verify the authenticity DNSKEY of the child zone by the DS record of its parent zone. How do we trust the parent zone? We trust the parent zone of the zone. Then there is a DNSSEC Trust Chain composed of DNS servers. If we trust root zone, which is believed to be carefully taken care of, then we can trust every DNS server on this trust chain.
   - Authenticated Denial of Existence: If we ping any website and the system responded, "unknown host", it may be the fake packet made by the attacker to purposely disable us from connecting to websites. To tackle this situation, DNSSEC sorts the DNS records lexicographically and put a digital signatured NSEC record between every two urls. When in queries of non-existed urls, DNSSEC will respond with the record of NSEC, and the DNS resolver use it to verify the non-existence of a record name and type as part of DNSSEC validation.

References

[1] en-Wiki DNS Spoofing
[2] en-Wiki Domain Name System Security Extensions
[3] The introduction of DNSSEC
[4] The basic principle of DNSSEC

DnsDoOmSday

1. There're a file 'query.log' and the 'bind' folder. I first checked the 'query.log' and observed that it recorded abnormal frequency of DNS query within 4 hours. Afterwards, I dug into the 'bind' folder. I took a while to figure out how it worked. Then, I launched my VM, installed bind9 and tried to diff the files in /etc/bind/ of my VM and Arvin's files. I found out that only two files are different, one is 'db.root', with trivial difference, and the other is 'named.conf.options'. In Arvin's latter file, I noticed that apart from some trivial settings, Arvin set up recursion yes to serve as a recursion server, and he also set the option, 'allow-recursion' to 'any'. To sum up, Arvin set up an open DNS revolver; consequently, it can lead to cache poison problem easily, and the worst case is being exploited to launch a DDoS attack.

References

[1] DNS: Open Resolver
[2] DNS: Resolving Process

2. There is a specific attack using open DNS revolvers called 'DNS amplification attack'. The attacker spoofs the source ip in DNS query packet with the victim's ip, then sends an enormous amount of DNS query(each packet is of small size) to the Open DNS server (in this case, Arvin's). The DNS server will in turn send back the DNS response packet(far bigger packet size than the query) to victim's IP. As a result, the target receives an amplification of the attacker's initial traffic, and the server is rendered inaccessible in such an attack.

References

[1] How to prevent the DNS server from being used to DDoS

[2] DNS Amplification Attack

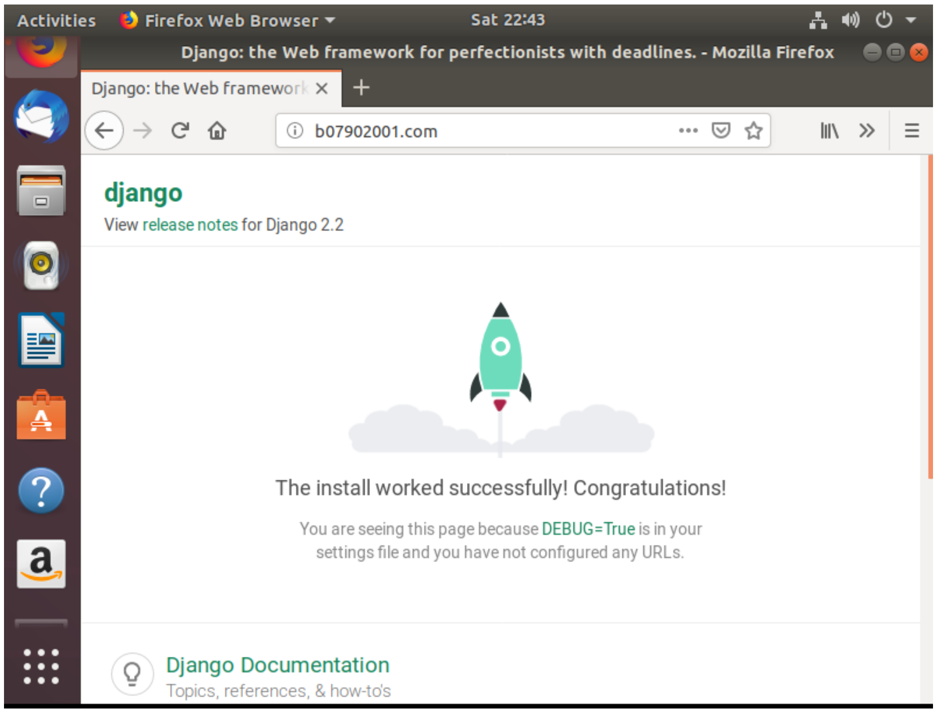
3. According to the log, the IP of DNS server is 168.95.98.200. There're 100 thousand lines of record, and the client IP is respective 140.112.30.0/24 (55.47%) and 168.95.98.0/24 (44.53%). From what I know about the DNS amplification attack, I doubt that all of the client IPs are spoofed by the attacker. Consequently, the DNS server will send the response packet back to those IPs, and causing DDoS of the two subnets.

4. He should filter his traffic using either firewall or configuration of bind9, allowing only trusted IPs. Bind9 has 'Access Control List' (ACL) to define a limited set of hosts in 'allow-query'.

References

[1] DNS: Open Resolver

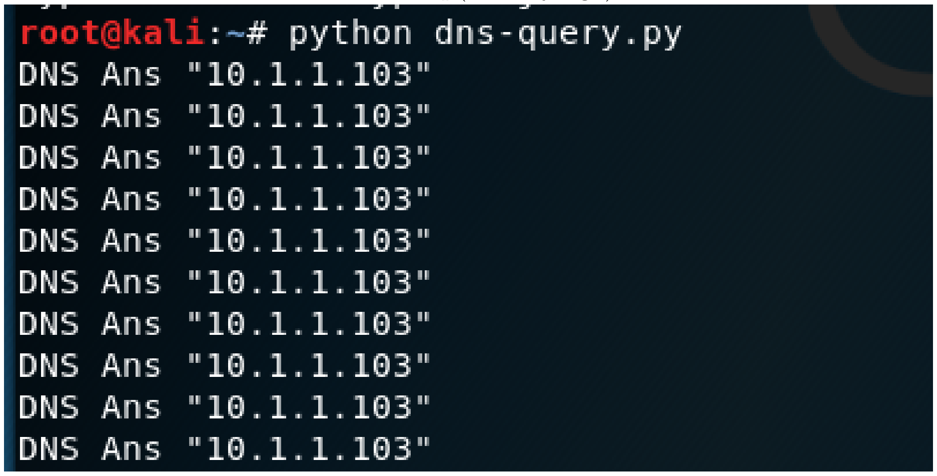
5. First,set up a web server in host (I use django in this case, and I don't think it necessary, just tried to make it closer to reality). Then set up the required files in the open DNS resolver. (Though troublesome, it's trivial in this problem, therefore I skipped explaining.) Afterwards, in Kali Linux, I created a python program using scapy based on someone's work. I spoofed my IP as the victim's IP and sent DNS query packet to the open DNS resolver; in turn, the DNS resolver sent back packet to the victim. The packets were captured in Wireshark, shown in below images. If needed, you can check my all files in this link.


Activities
Firefox Web Browser
Sat 22:43
Django: the Web framework for perfectionists with deadlines. - Mozilla Firefox
Django: the Web framework
b07902001.com
django
View release notes for Django 2.2
Django Documentation
Topics, references, & how-to's

The install worked successfully! Congratulations!

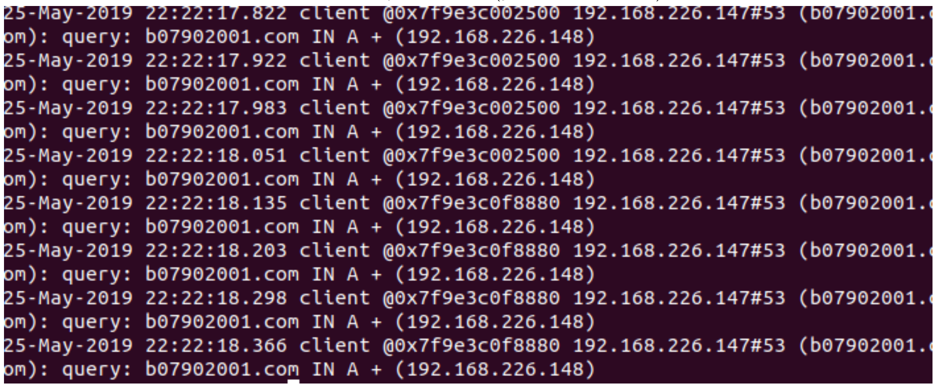
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

DNS set up(with django)



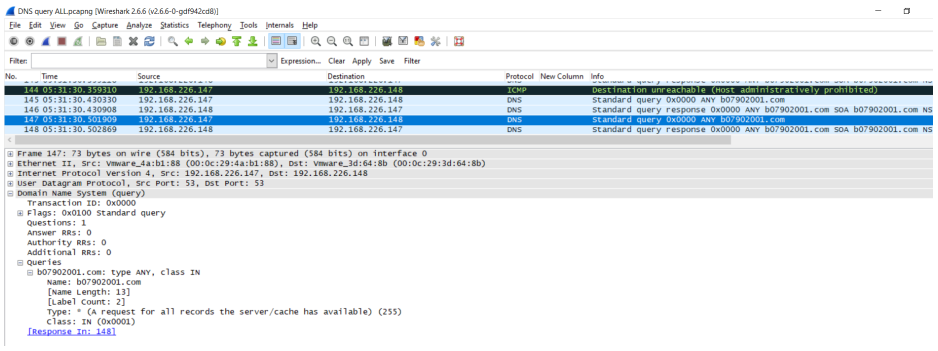
```
root@kali:~# python dns-query.py
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
DNS Ans "10.1.1.103"
```

DNS query in kali (just A record)



```
25-May-2019 22:22:17.822 client @0x7f9e3c002500 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:17.922 client @0x7f9e3c002500 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:17.983 client @0x7f9e3c002500 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:18.051 client @0x7f9e3c002500 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:18.135 client @0x7f9e3c0f8880 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:18.203 client @0x7f9e3c0f8880 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:18.298 client @0x7f9e3c0f8880 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
25-May-2019 22:22:18.366 client @0x7f9e3c0f8880 192.168.226.147#53 (b07902001.
om): query: b07902001.com IN A + (192.168.226.148)
```

Query log in the open DNS resolver

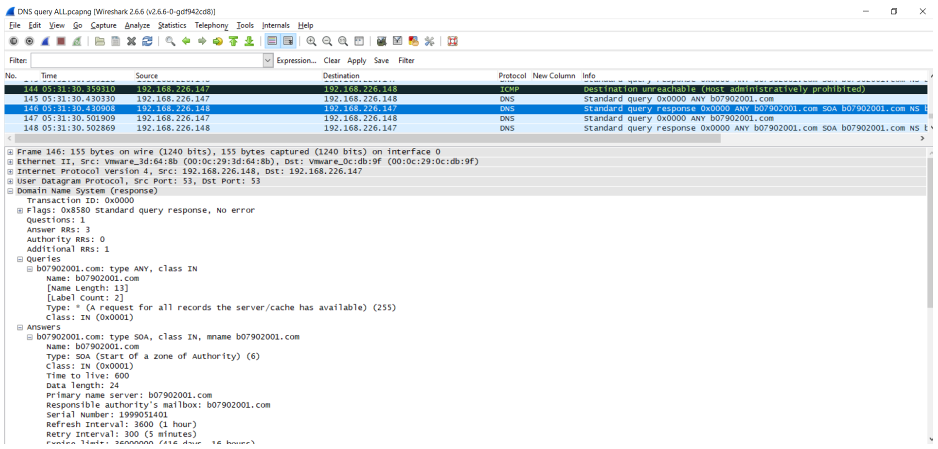


```
DNS query ALL.pcapng [Wireshark 2.6.6 (v2.6.6-0-gdf942cd8)]
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Tools  Internals  Help
Filter:                                   Expression...  Clear  Apply  Save  Filter

No.  Time             Source            Destination       Protocol  New Column  Info
144 05:31:30.359310   192.168.226.147   192.168.226.148   ICMP                  Destination unreachable (Host administratively prohibited)
145 05:31:30.430330   192.168.226.147   192.168.226.148   DNS                   Standard query 0x0000 ANY b07902001.com
146 05:31:30.430908   192.168.226.148   192.168.226.147   DNS                   Standard query response 0x0000 ANY b07902001.com SOA b07902001.com NS
147 05:31:30.501909   192.168.226.147   192.168.226.148   DNS                   Standard query 0x0000 ANY b07902001.com
148 05:31:30.502869   192.168.226.148   192.168.226.147   DNS                   Standard query response 0x0000 ANY b07902001.com SOA b07902001.com NS

⊞ Frame 147: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
⊞ Ethernet II, Src: Vmware_4a:b1:88 (00:0c:29:4a:b1:88), Dst: Vmware_3d:64:8b (00:0c:29:3d:64:8b)
⊞ Internet Protocol Version 4, Src: 192.168.226.147, Dst: 192.168.226.148
⊞ User Datagram Protocol, Src Port: 53, Dst Port: 53
⊟ Domain Name System (query)
    Transaction ID: 0x0000
  ⊞ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊟ b07902001.com: type ANY, class IN
        Name: b07902001.com
        [Name Length: 13]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
    [Response In: 148]
```

DNS query packet in Wireshark



```
DNS query ALL.pcapng [Wireshark 2.6.6 (v2.6.6-0-gdf942cd8)]
File  Edit  View  Go  Capture  Analyze  Statistics  Telephony  Tools  Internals  Help
Filter:                                   Expression...  Clear  Apply  Save  Filter

No.  Time             Source            Destination       Protocol  New Column  Info
144 05:31:30.359310   192.168.226.147   192.168.226.148   ICMP                  Destination unreachable (Host administratively prohibited)
145 05:31:30.430330   192.168.226.147   192.168.226.148   DNS                   Standard query 0x0000 ANY b07902001.com
146 05:31:30.430908   192.168.226.148   192.168.226.147   DNS                   Standard query response 0x0000 ANY b07902001.com SOA b07902001.com NS b
147 05:31:30.501909   192.168.226.147   192.168.226.148   DNS                   Standard query 0x0000 ANY b07902001.com
148 05:31:30.502869   192.168.226.148   192.168.226.147   DNS                   Standard query response 0x0000 ANY b07902001.com SOA b07902001.com NS b

⊞ Frame 146: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface 0
⊞ Ethernet II, Src: Vmware_3d:64:8b (00:0c:29:3d:64:8b), Dst: Vmware_0c:db:9f (00:0c:29:0c:db:9f)
⊞ Internet Protocol Version 4, Src: 192.168.226.148, Dst: 192.168.226.147
⊞ User Datagram Protocol, Src Port: 53, Dst Port: 53
⊟ Domain Name System (response)
    Transaction ID: 0x0000
  ⊞ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 1
  ⊟ Queries
    ⊟ b07902001.com: type ANY, class IN
        Name: b07902001.com
        [Name Length: 13]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
  ⊟ Answers
    ⊟ b07902001.com: type SOA, class IN, mname b07902001.com
        Name: b07902001.com
        Type: SOA (Start Of a zone of Authority) (6)
        Class: IN (0x0001)
        Time to live: 600
        Data length: 24
        Primary name server: b07902001.com
        Responsible authority's mailbox: b07902001.com
        Serial Number: 1999051401
        Refresh Interval: 3600 (1 hour)
        Retry Interval: 300 (5 minutes)
        Expire limit: 36000000 (416 days, 16 hours)
```

DNS response packet in Wireshark

References

[1] Vbird: DNS server

[2] NASA DNS Lab

[3] thepacketgeek/10-dns-query.py