Bachelor Thesis

Analysis of authoritative DNS infrastructure failures

Ege Girit

December 24, 2022

Reviewer:
Prof. Dr. Christian Rossow
BSc. Jonas Bushart

Contents

1 Introduction

1.1 Structure Of The Work

This bachelor thesis is divided into four main parts. In the first part, a summary of the Domain Name System is provided. Building on this, we explain why we believe that there is a need for research into resolver behavior during an authoritative server failure. The second part details the characteristics we want to examine, the test sources, and the metrics for attack success. This part also presents the test environment and describes the experiments and analysis methods. The results obtained from our experiments are shown in part three, supported by visual graphs. These results are then evaluated in part four to draw conclusions about what the tests show. Finally, it is discussed whether and how much our results deviate from reality and whether more research is needed in this area.

1.2 Domain Name System

The need for the Domain Name System arose from the need to address the IP addresses of servers, which are difficult for humans to remember, with easy-to-remember names. DNS can be used to resolve the names of a server to their associated IP address.
Before the Domain Name System, there was only a centralized text file (hosts.txt) consisting of static mappings that allowed users to resolve stored web pages to their associated IP addresses. This text file had to be updated after each change and distributed to all computers to provide a mapping between these names and their corresponding network addresses. With the growing number of hosts and websites, it became impractical to manage such a centralized system. The administrative overhead associated with managing every possible domain name on the Internet would be too great, and this central database would not scale well. Faced with these challenges, DNS designers moved away from the flat naming approach and adopted a decentralized model with a hierarchical naming architecture that became the Internet standard in 1986. The official documentation of these standards is described in RFC 1034 and 1035. However, the hosts.txt file still exists for various purposes like performance improvement and URL filtering.

Today we have a globally distributed collection of databases that form a tree structure. System administration is completely decentralized, and delegation is handed over to organizations. There are three main functions of DNS:

1. Namespace (Checking whether the syntax and structure of the domain is valid)
2. Name registration (Whether the name to be registered is unique)
3. Name resolution

In this work, we will focus on the name resolution function of DNS. Several actors play a role in resolving names to their corresponding IP addresses. These actors are:

- The client
- Recursive resolver
- Root name server
- TLD Server
- Authoritative name server

The client is interested in the answer of the name resolution. To start the name resolution process, a client sends a DNS query to a recursive DNS resolver. The DNS resolver then takes the role of the client and starts the querying process by trying to find the answer to the query on behalf of client and returning the results to the client. The recursive resolver acts as a middleman between the client and the DNS hierachy. The recursive resolver contacts the DNS hierarchy to get the DNS response. The first server that the recursive resolver contacts is the root name server.
The root name server sends back the appropriate top-level domain (TLD) server based on the queried host's domain extension to the recursive resolver. The TLD server receives the next request from the recursive resolver and responds with the appropriate authoritative nameserver. The authoritative nameserver stores the DNS records that map domain names to IP addresses, and it responds to the recursive resolver's final request with the queried hostname's IP address. If the IP address is not available, the nameserver will throw an error.
If the resolution process takes too much time, the resolution times out. Because there are many servers involved in this process, a request from a client can consume a lot of network resources before the resolution succeeds or fails. Therefore, there are optimization mechanisms such as caching, which shorten the resolution time and save resources. The purpose of caching is to temporarily store previously requested data so that future requests for that data can be served faster without contacting all servers in the DNS hierarchy.
A DNS retransmission is the process of sending a DNS query again if no response or an error response is received from the DNS server. DNS retransmissions are typically triggered by a timeout, which occurs when the DNS server does not respond within a certain time frame. The client will typically retransmit the query multiple times before giving up and returning an error to the user.

For a DNS request, there can be multiple duplicate requests, and therefore multiple responses to these requests. If a resolver has not received an answer to a query, the resolver can repeat this query, resulting in duplicate queries and possible duplicate responses. For a DNS request, we calculate the latency by calculating the time difference between the first DNS query and the first DNS response to it, which has the RCODE OK, meaning that the DNS resolution was successfull.

1.3 Motivation

DNS is a fundamental protocol that was not designed with security in mind. There are many types of attacks aimed at exploiting vulnerabilities in DNS. If name resolution does not work properly, there is no alternate protocol for name resolution, and the connectivity between users and the server is compromised. This makes DNS a popular target for hackers. Many systems rely on DNS to remain functional. Therefore, errors or attacks on DNS are of particular importance for everything based on it. When DNS goes down, the Internet goes down. If an authoritative name server is unreachable, the client usually has to expect longer waiting times or errors. However, we lack knowledge about resolver behavior during a cyber attack and authoritative server failures.

In this thesis, we examine the behavior of resolvers in the event of a (partial or complete) failure of critical authoritative DNS servers. Our task is to understand from the results of our experiments whether the behavior of resolvers follows certain patterns. The findings of this behavior analysis could help us better understand the influence of individual parameters when communicating with the name server and to predict the behavior of the resolver.

Similar research has been done in this area where the client's point of view is mainly examined in the process of DNS resolution. Our work is different in that we focus on the role of the recursive resolvers and examine their behavior in our experiments.

2 Structure of the experiment

2.1 Parameters To Be Tested

In our experiments, we want to know whether the simulated cyber attack against the DNS defense mechanisms is successful. For this we measure the following metrics:

- DNS resolution latency, which is the time it takes for a DNS resolver to receive a response from a DNS server in response to a DNS query
- Failure rate to examine the percentage of failed DNS packets
- The underlying transport protocol of the DNS packet (either UDP or TCP)
- Rate of stale records to examine whether a resolver is able to answer a client query in case of a name server failure
- DNS retransmissions to examine how the packetloss amplifies the DNS traffic by resending a DNS query

2.2 Test Sources

Since there are many resolvers with various configurations, we want to carry out our experiments in such a way that we have the greatest possible coverage of all types of resolvers in order to determine a generalization of the resolver behavior. The types of resolvers we tested in our experiments are as follows:

- Widely used open resolvers
- Resolver of RIPE atlas probes
- Wild open resolvers found via DNS scans

For our experiments, we looked for widely used DNS providers/operators. The DNS operators we have chosen are the following: AdGuard, CleanBrowsing, Cloudflare, Dyn, Google, Neustar, Norton, Level3, OpenDNS, Yandex and Quad9. We examined a total of 11 operators and 30 open resolver IP addresses in our experiments. When choosing the IP addresses, we made sure that we have at least one resolver IP address for each configuration of the operator, so that we can test all different configurations from one operator. In our stale record experiments we show that different configurations of the same operator behaves differently in case of a DNS server failure.

RIPE Atlas is a global network of probes that help us measure the internet connectivity and reachability in real time. The probes are small devices that are distributed around the world and connected to the Internet through various types of connections. Each

probe sends measurements to the RIPE Atlas servers and the collected data is used to provide a detailed view of the Internet's performance, including metrics such as latency, packetloss, and throughput. The data collected by RIPE Atlas is used to understand the performance and reliability of the Internet.

2.3 Details Of The Test Environment

For our experiments we use an authoritative name server at Saarland University, that uses BIND9 as an open-source name server software to be able to interact with DNS. We only use one name server with one IP address without the use of any anycast structure or secondary servers. The configuration files of the servers that we used in our experiments can be found in the experiment repository.

iptables is a command-line utility in Linux that allows to configure the tables provided by the Linux kernel firewall and the chains and rules it stores. iptables uses a set of tables, which contain chains of rules that match packets and determine what to do with them. It can be used to set up firewall rules that allow or block traffic based on various criteria, such as the source or destination IP address, the protocol, and the port number. To simulate an attack on our authoritative server, iptable rules are used by configuring it to randomly drop incoming UDP and TCP packets on the server at a specified rate. Our authoritative name server acts as both a server, providing DNS answers to a resolver, and a client, creating the DNS queries. So that we can differentiate between client and server network traffic, we use different IP addresses for the client and the server.
We use the DNSPython library for creating and sending DNS requests. All the DNS queries sent in our experiments use a default timeout value of 10 seconds. The command-line utility tcpdump is used on our authoritative server to capture and display incoming and outgoing network packets transmitted over the network such as DNS queries and responses. To isolate the recorded network traffic for the respective resolver communications, we use wireshark to filter irrelevant packets, that are not generated by our experiment. After we have performed the experiments, the stored capture files are evaluated and the data of interest is extracted and visualised by the use of various plots to recognize the results.

3 Experiments and Results

3.1 Open Resolver Packetloss Experiment

Since we cannot perform a real cyber attack on our authoritative name server, we simulate a DDoS attack by applying iptable rules that introduce random packetloss for incoming packets on our server with a specified packetloss rate.
DDoS stands for Distributed Denial of Service and it is a type of cyber attack in which a large number of computers, which are often compromised, are used to flood a target network or website with network traffic in an attempt to make it unavailable to users. A DDoS attack can create a high network load on the target server by flooding it with a large volume of traffic, which can make it difficult or impossible for the server to process and respond to legitimate requests. As a result, some network packets may be dropped or may not be sent successfully, leading to a partial or complete disruption of service for legitimate users. For our experiment we use various rates of packetloss and these packetloss rates are 10, 20, 30, 40, 50, 60, 70, 80, 85, 90 and 95.
To avoid caching on the resolver side and measuring the latency of the full query lookup without any optimization mechanism, we used a unique query name for each query we sent to the resolver. Without caching, the resolver would have to contact the DNS hierarchy to look up the corresponding IP address to the query.
(how the retransmissions help when packetloss active, paper)
Per packetloss rate we sent 49 queries to each resolver. It is generally not appropriate to send a large number of queries to a DNS resolver in a short period of time, as this can put a strain on the resolver and potentially impact its performance for other users. There is no specific threshold for how many queries is considered "excessive", as it can vary depending on the capabilities of the DNS resolver and the overall level of activity on the network. In this experiment, we introduced a one second delay after each query sending. After Sending 49 queries to a resolver for a packetloss rate, we added a cooldown phase of 10 minutes, where we kept the network capture going to be able to see any delayed dns packets. After the cooldown phase, we continued the experiment with the next packetloss rate, until all the packetloss rates are processed.

3.2 Open Resolver Packetloss Experiment Results

To visualise the observed latencies, we use violin plot, which allows us to see the range as well as the density of the latencies. The thicker part of the plot represents the denser regions of the data, while the thinner part represents the sparser regions of the data.

3.3 Ripe Atlas Packetloss Experiment

Due to the daily kredit limit of the ripe atlas measurement system, we carried out our packetloss experiment with the ripe atlas in 2 days, with the first day including the packetloss rates 40, 60, 70, 80, 90, 95, and the second day including the packetloss rates 0, 10, 20, 30, 50, 85. In total we experimented with 830 worldwide uniform distributed probes.

3.4 Ripe Atlas Packetloss Experiment Results

3.5 Wild Open Resolver Experiment

3.6 Wild Open Resolver Experiment Results

3.7 Stale Record Support Experiment

A DNS request from a client can consume a lot of network resources until the resolution succeeds or fails. Therefore, there are optimization mechanisms such as caching, which shorten the resolution time and save resources. The purpose of caching is to temporarily store previously requested data so that future DNS requests for that data can be served faster without contacting all servers in the DNS hierarchy. The resolver stores the cached entries and these cached entries have a TTL (Time to live) value. This value represents the number of seconds that the resolver should keep the entry in its cache. However, in practice, the resolver may alter this TTL value based on its implementation, resulting in the entry being kept for more or less time. According to the studies made by ..., the resolvers usually shortens the TTL time rather than keeping the data more than the TTL value. A lower TTL value means that the lookups for the resolutions are needed to be performed more frequently.

If the resolver detects that the authoritative name server is unreachable, then the resolver may respond the client with old records. We wanted to measure how widespread this strategy is among resolvers. A DNS record, that is stored in a resolvers cache is considered stale, when that records TTL value is expired. Many resolvers clear the expired entries from their caches automatically to free the storage space. The deletion of stale records depends on the implementation of the cache data structure and the configuration of the resolver. The calculation of whether a record is stale in BIND9 only happens during the lookup dynamically. BIND9 does not know which lookups expire when and it uses a hash table to store and manage stale records.

Besides its advantages for static/long term IP addresses, stale DNS records can also cause issues with the resolution of domain names for IP addresses that change often, leading to errors or incorrect information being returned to users. If a DNS record is associated with an IP address that is no longer in use or has been reassigned to a different

domain, the DNS system may still return the old IP address when a user attempts to access the domain. This can cause users to be directed to the wrong website or to receive an error message when trying to access the domain.

The goal of our first stale record experiment is to determine if a resolver is able to keep answering the client queries with stale records in case of total unavailability of the authoritative server, even though the resolvers cache entries are expired. This experiment also allows us to see if stale records can help in case the server has an outage.

Each resolver IP address can have a different number of caches and the resolvers might be deployed behind a load balancer architecture to distribute the network load. The implementation of the load balancer influences the decision, which server should answer the incoming DNS query. With this architecture, each individual server might maintain a cache that is isolated from the other servers. To increase the cache hit rates, some architectures might use a shared cache pool that can be used by multiple servers.

Because of the varying cache counts of each recursiver resolver, we have to treat each resolver differently and be certain that we have first filled all the caches of a resolver with an entry to be able to create stale record entries at the resolver side.

We divide our first stale record experiment in two phases, namely the prefetching phase and the stale phase. In the prefetching phase, we create a DNS query and send it to the DNS resolvers so many times, so that all of the caches of the resolver is filled with a 95% chance, assuming the load balancers use a uniformly distributed implementation. We want to fill all the caches of the resolver with this query because if a cache is not hit, and we send the same DNS query to the resolver in the stale phase and hit an unfilled cache, the resolvers response will (most likely) be a resolution error. If we miss a resolvers cache, a stale record for that missed cache can't exist, therefore we cant test if the resolver supports stale records or not. One of the challenges of this phase is the unknown cache count of the resolvers. Every resolver can have different cache counts and these cache counts could be implemented in a way, so that it varies during the day based on the network load of the resolver. To estimate the cache counts of each DNS operators resolvers, we did separate experiments at different times of the day and we took the maximum cache counts of these experiment as result. We then started our stale record experiment with the found cache amounts for each resolver.

To understand if a resolver is really sending a stale record and not fetching a fresh record from our server, we changed the IP addresses of the DNS responses on our server after the prefetching phase, so that we can examine the received response on the client side using this IP address, if the answer of the resolver was stale. (Diagram for this?) After filling all caches of all resolver IP addresses in the prefetching phase, we simulated the selected packetloss rates on our authoritative name server. With the simulated packetloss rate, we started making DNS requests to the resolvers again. We then examined the DNS responses and the results of this experiment.

3.8 Stale Record Support Experiment Result

The first stale record experiment allowed us to divide the selected resolvers we are investigating into 2 groups. On the one hand resolvers who like to send stale records, and on the other hand resolvers that only send up to a few stale/single records. For these 2 groups we then compared the results to see how the stale records affected the DNS resolution.

We received stale records from only 6 of the 30 resolver IP addresses we investigated. We observed that a dns operator might make use of stale records for only some of their IP addresses and some of the IP addresses might not use stale resords at all. We selected 3 resolver IP addresses for the operator OpenDNS and 2 of them sent stale records, while one of the IPs never sent a stale record. (Decision of taking 1 IP pro config was useful, different configs may behave differently in case of failure)
... (1. stale experiment plots)

When we compare the failure rates of these two groups, we clearly see that the query resolution process benefit from the stale records. While stale records can be useful for IP Addresses that doesn't change often, A stale DNS record is no longer accurate or relevant when a domain name is transferred to a new owner, when a website's IP address changes, or when a resource is no longer available.

3.9 Stale Record TTL Experiment

After the first stale record experiment, we wanted to test, for how long does the resolver sends stale records, if the resolver is able to send stale records. If we answer this question, we can estimate for how long a resolver can provide DNS responses, as if there is no failure on the name server at all.

3.10 Stale Record TTL Experiment Result

... (duration plots)

TTL Experiment...

3.11 Stale Record Send Frequency Experiment

constant frequency experiment..

3.12 Stale Record Send Frequency Experiment Result

constant frequency experiment results..

3.13 DNS Truncation Experiment

IP spoofing is a type of cyber attack in which an attacker alters the source IP address of a packet to make it appear as if it came from a different device. This is typically done to hide the identity of the attacker, or to launch attacks on networks or individual devices.
"IP-Spoofing" in the context of networking means that the communication appears to originate from another host by manipulating IP packets with a forged sender IP address.
DNS Cookies work by adding an additional layer of protection to the DNS protocol, allowing DNS servers and resolvers to authenticate each other and verify the integrity of DNS responses. DNS cookies use cryptographic techniques to create a small piece of data that is sent between a DNS resolver and a DNS server. Although they are not a complete solution for all security problems, they provide a lightweight mitigation mechanism against attacks such as reflection attacks and DDoS attacks from spoofed sources. In order to benefit from DNS Cookies and decrease their vulnerability to the threats, both the client and server must support/implement it. Another way to protect against IP-Spoofing attacks is by using TCP.
TCP uses a 3-way handshake to initiate a connection between the sender and receiver, which uses a random number that prevents IP-Spoofing. Due to the sequence number mechanism of the TCP connections, this IP address manipulation via TCP on an existing connection is only possible by bruteforcing the sequence number or eavesdropping on the client, which makes TCP much more secure than UDP. UDP has no sequence numbers, therefore it is possible to insert manipulated data that be treated as if they were coming from the real host.
DNS queries are usually sent via the UDP protocol and as a result, it is possible to send a forged DNS query with a spoofed IP address to a resolver.

A response policy zone (RPZ) in BIND9 is a feature that allows the DNS server to modify its responses to DNS queries based on a set of rules. The RPZ zone is a special type of zone that contains the rules for modifying DNS responses. These rules can be used to block access to certain domains, redirect traffic to different domains, or provide alternate responses for specific queries. The RPZ zone can be stored in a separate file on the DNS server, or it can be included in the server's main configuration file. Once the RPZ zone is configured, the DNS server will apply the rules in the RPZ zone to all DNS queries that it receives. If a query matches one of the rules in the RPZ zone, the DNS server will modify its response accordingly.
(Paper notes: what when cookies enforced, support in world...)
Our packetloss experiment showed that none of the resolver we selected switched their communication channel from UDP to TCP by themselves. Only with truncation experiment we saw TCP packets. (what happens if the server dictates that from now on it will only be addressed completely via TCP. If a DNS server is attacked or similar, it may be that these packages come from spoofed IP addresses. This is not entirely visible to auth server. If you counted the number of IP addresses, you would probably see that, but you have to look straight after it. Spoofed packets come via UDP. TCP is not spoofable

because of the 3-way handshake. What you could test here would be what happens if we don't accept UDP packets. We don't discard these UDP packets, but DNS has mechanism to indicate that the response was incomplete, by setting the truncation bit at the header (tell the resolver that the response was too large, try TCP).
(Switch-over rate to TCP) (With truncation, the server dictates that from that point onwards it will only be addressed completely via TCP. This is important to ensure that the resolver always gets answers.)

3.14 DNS Truncation Experiment Results

4 Conclusions/Evaluation

...

5 Discussion

Only PC users tested, mobile/other devices? ...

6 Ethical Considerations

All the experiments we introduced in this paper are performed in a benign manner so that our experiments does not cause loads of network traffic on the resolver side. The highest frequency that the queries are sent to a resolver was the prefetching phases of the stale records experiments. In the prefetching phase, we have sent 2 queries every second to a resolver with a high amount of cache count for about 30 seconds to be able to hit all the caches of the resolver in a short time.

Bibliography

Eidesstattliche Versicherung

(Affidavit)

Name, Vorname
(surname, first name)

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☐ Masterarbeit
(Master's thesis)

Titel
(Title)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**