Bachelor Thesis

Analysis of authoritative DNS infrastructure failures

Ege Girit

December 4, 2022

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

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl V - Programmiersysteme
Fachgruppe Softwaretechnik sicherer Systeme
https://sse.cs.tu-dortmund.de

Contents

1 Introduction

1.1 Structure Of The Work

This bachelor thesis is divided into 4 main parts. The first part gives a summary of the Domain Name System. Building on this, it is then explained why we believe that there is a need for research into resolver behavior during an authoritative server failure. The characteristics of what we want to examine, the possibilities of the test sources and metrics for attack success are given in Part 2. The test environment is presented and the course of the experiments and analysis methods are made clear. The results obtained from our experiments are shown in part 3 supported by the usage of various visual graphs. These results are then evaluated in Part 4 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 Domain Name System arose from the need to address the IP addresses of servers, which are difficult for humans to remember, by an easy-to-remember name. DNS can be used to resolve the names of a server to their associated IP address.

Before Domain Name System, there was only a centralized text file (hosts.txt) consisting of static mappings that allowed users to resolve the 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 is now impossible 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 3 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 name resolution in 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 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. A 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 a recursive resolvers 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.

1.3 Motivation

DNS is a fundamental protocol and was not designed with security in mind. There are many types of attacks aimed at exploiting vulnerabilities in DNS. If the name resolution does not work properly, there is no alternate protocol for the name resolution and the connectivity between the 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, the behavior of resolvers in the event of a (partial) failure of critical authoritative DNS servers is examined. Our task is to be able to understand from the results of our experiments whether the behavior of resolvers follows certain patterns. The findings of this behavior analysis could help to better understand the influence of the individual parameters when communicating with the name server and to predict the behavior of the resolver.

Similar researches were made in this area where the client point of view is mainly examined in the process of a DNS resolution. Our work is different in a way that we focus on the role of the recursive resolvers and examine their behaviour in our experiments.

2 Structure of the experiment

2.1 Parameters To Be Tested

For our experiment, we dont use anycast.
Things to test: Various rates of packet loss, truncation (switch to TCP), DNS Cookies, RRL (no response vs truncation), DNSSEC NSEC3 caching, stale records. Overall we did 6 different experiments. These are:

- Simulating packet loss and measuring the DNS resolution latencies and resolution failure rate
- Simulating packet loss and measuring if the resolver uses stale records for the resolution
- ...

2.2 Metrics For An Attacks Success

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 response failure rate (Client side), DNS response rate (Server side), DNS response latency, Query duplicates, Switch-over rate to TCP, Rate of stale records, Name server migrations (and reasons thereof), Multiple NS records, One NS with multiple A/AAAA records, (Anycast), Page load time (Web context), Attack amplification,

2.3 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:

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

RIPE Atlas is a global network of probes that help us measure the Internet connectivity and reachability in real time. Due to the daily kredit limit of the ripe atlas measurement system, we carried out our experiments with the ripe atlas in 2 days, with the first day including the packet loss rates 40, 60, 70, 80, 90, 95, and the second day including the

packet loss rates 0, 10, 20, 30, 50, 85. In total we experimented with 830 worldwide distributed probes.
For our experiments, we looked for widely used DNS providers/operators. The DNS operators we have chosen are the following: AdGuard, Alternate DNS, Control D , CleanBrowsing, Cloudflare, Dyn, Google, Neustar, OpenDNS, Yandex and Quad9. We examined a total of 11 operators and 31 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.

2.4 Latency And Failure Rate Experiments

Since we cannot perform a real cyber attack on our authoritative name server, we will simulate a DDoS attack by simulating random packetloss for incoming packets to our server with a given rate on our server. A DDoS attack creates a high network load on the server and this means that some network packets cannot be sent successfully. We use different rates of packetloss (10, 20, 30, 40, 50, 60, 70, 80, 85, 90, 95) for our experiments. In our experiment we measure the rate of TCP truncations, the DNS resolution latencies, the failure rate of the DNS resolution and the dns retransmission count. 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.
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.

2.5 Stale Record Experiments

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. TTL is the time in seconds that a resolver should keep that entry in its cache. But in practice it can happen that the resolver changes this TTL value depending on its implementation and keeps the entry 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 time to live value (TTL) is expired. Many resolvers clear/flush the expired entries from their caches automatically to free the storage space. The goal of the stale record experiments is to determine if a resolver keeps answering the client queries with stale records, in case of unavailability of the authoritative server. The deletion of stale records depends on the implementation of the cache data structure and the configuration of the resolver.
The calculation of whether something is stale in the Bind software only happens during the lookup dynamically, bind does not know which lookups expire when, bind has a hash lookup table based on query name and query type/class.

The stale record experiment is about testing whether the resolvers we have chosen are able to send such stale records. Our goal is to check if, even though the resolvers cache entries are expired, they're still sending these stale entries. This experiment also allows us to see if stale records can help with this in case the server has an outage.
The first stale record experiment allowed us to divide the 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.

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 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.
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 to our server after the prefetching phase, so that we can examine the response using this IP address in the response, if the answer of the resolver was stale. 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 allowed us to divide the selected resolver IP addresses we are examining into 2 groups. These 2 groups are; resolvers that can send stale records, and resolvers, that don't send any stale records at all. For these 2 groups we then compared the results to see how the stale records affected the DNS resolution.
... (1. stale experiment plots)

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.
... (duration plots)

2.6 DNS Cookies Experiment

"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. 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 using port 53 (?). As a result, it is possible to send a forged DNS query with a spoofed IP address to a resolver.

DNS Cookies: DNS Cookies provide protection for the clients, domain name owners , the innocent and DNS Servers. It is a lightweight mitigation and not a complete solution for all security problems. There are certain attacks they don't protect against. The client who makes a DNS query can get a false answer/response. If an attackers answers the client query before the legitimate DNS server, attackers response is accepted on

the client side (if client is a recursive resolver -> cache poisoning.) Cookies protect an innocent victim against spoofed IP address attacks. An attacker can forge a DNS query using a victims IP address, and the DNS Servers will respond to the victims IP address (The response will be bigger than the query because of UDP(amplification), many dns servers will contact the victim IP like a botnet -> Reflection attack). An authoritative name server can protect itself from DDoS attacks, they identify spoofed IP's and the server limits its misuse in a reflection, responding with TC bit (TCP). ... If only one party to a DNS transaction supports DNS Cookies, the mechanism does not provide a benefit or significantly interfere, but if both support it, the additional security provided is automatically available. ...
Although DNS Cookies 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. IF DNS Cookies are not implemented at a server, the server responds as if ...?

2.7 Details Of The Test Environment

We set up our own name server at Saarland University (to simulate the DNS infrastructure). For this purpose, an authoritative name server is configured on a Linux computer. To be able to interact with DNS, we use the BIND9 software. The configuration files of the servers that we used in our experiments are available/can be found in our repository. With the use of iptables, where we can specify the desired packetloss rate, an attack on the server is simulated by configuring random packetloss on the incoming packets of the server, which drops the incoming packets randomly. 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. The linux software tcpdump is used on our authoritative server to store incoming and outgoing network packets 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 saved log files are evaluated using a script, which creates plots to visualize the experiment results. In order to be able to visualize the data, the matplotlib library is used. The saved log data with a PCAP format is converted into JSON format and the parts of interest are extracted.

3 Experiment Results

3.1 Open Resolver Packet Loss Tests

3.2 Ripe Atlas Packet Loss Tests

3.3 Stale Records Tests

3.4 DNS Cookies Tests

3.5 Truncation Tests

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.**