TECHNISCHE HOCHSCHULE LÜBECK

Ausarbeitung

Funktionsweise des Domain Name System - eine Betrachtung

Vorgelegt von: Jan-Ole Hübner

Inhaltsverzeichnis

1 Einleitung

Das *Domain Name System* (kurz DNS) ist aus dem Internet nicht mehr wegzudenken. Es ist eine der großen Technologien auf denen das ganze Netz aufbaut. Es ist sozusagen das Telefonbuch der Moderne. Mittlerweile ist aus dem Handel mit Domain Namen eine ganze Branche entstanden. Allein Die DENIC eG verzeichnet - stand Januar 2021 - 16,7 Millionen Domains [eG21] mit der Endung .de. Doch kaum jemand macht sich Gedanken darum wie dieses essentielle System arbeitet oder was es eigentlich tut. Diese Arbeit soll die Funktionsweise des *Domain Name System* erläutern und den aktuellen Stand der Technik darstellen, sowie in hinblick auf die Sicherheit betrachten.

1.1 Was ist DNS?

Was ist also DNS? Kurz gesagt kümmert sich das *Domain Name System* um die Übersetzung von Domain Namen in IP-Adressen. Gibt ein Benutzer in einem Browser beispielsweise die Adressen *jan-ole.de* ein, kann dieser mit der Zeichenkette erst einmal nichts anfangen. Zur Kommunikation mit dem Webserver wird eine IP-Adresse benötigt. Genau diese Aufgabe übernimmt das DNS. Es 'übersetzt' den Namen in eine IP-Adresse. Abbildung 1.1 zeigt

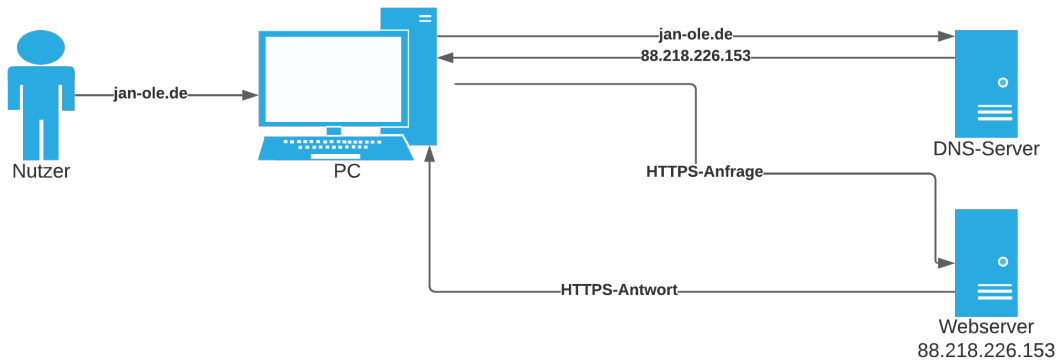

Abbildung 1.1: Überblick DNS

einen groben Überblick des Ablaufs. Der Benutzer gibt die Adresse ein und der Name wird mittels einer Anfrage an einen DNS-Server *aufgelöst*. Das bedeutet, der DNS-Server antwortet mit der IP-Adresse des dazugehörigen Servers. Daraufhin kann die IP-Adresse für weitere

Kommunikation verwendet werden, ohne dass der Benutzer diese eingeben musste.
Abbildung 1.1 stellt den Ablauf jedoch sehr stark vereinfacht dar. Vieles wird hier nicht berücksichtigt. Woher hat der DNS-Server die Information? Wie stellt er deren Richtigkeit sicher? Und wie kann diese Information geändert werden? Damit beschäftigt sich Kapitel 2.

1.2 Wie ist DNS entstanden?

Zu Zeiten des ARPAnets in den 1970er bestand das Netz aus wenigen hundert Hosts, die Zuordnung von Namen auf Adressen erfolgte in einer einfachen Textdatei Namens 'HOSTS'. [LA06] Als das Netz später auf TCP/IP umstellte stieg die Anzahl der Hosts rasant an. Daraus ergaben einige Probleme. Zum einen entstand aufgrund der steigenden Anzahl an Hosts durch das Verteilen der HOSTS Datei eine große Menge an Datenverkehr, zum Anderen gab es aufgrund der Größe des Netzes Konsistenzprobleme bei der Verteilung der Datei. Auch Kollisionen bei der Namensauflösung aufgrund doppelt verwendeter Namen wurden zu einem Problem. Ein Nachfolger für die HOSTS Datei - die Heute noch in den gängigen Betriebssystemen zu finden ist - musste entwickelt werden. Es sollte dezentralisiert und hierarchisch aufgebaut sein.
In 1984 veröffentlichte Paul Mockapetris die RFCs 882 und 883, die das *Domain Name System* beschreiben und seither mehrere Male aktualisiert und erweitert wurden. [Moc83] Beispielsweise definiert RFC920 die Anforderungen an eine Domain und legt fest, dass das Management der Namensräume aufgeteilt werden kann. Hier wurden die *Top Level Domains* .arpa, .com, .edu, .gov, .mil und .org festgelegt. [Pos84]

2 Architektur

2.1 Namespaces

Das *Domain Name System* ist eine hierachisch strukturierte, verteilte Datenbank, die Ihre Daten im gesamten Netwerk zur verfügung stellt. Strukturiert ist das DNS ähnlich eines UNIX-Dateisystems in einem Baum. Jeder Knoten des Baums steht hierbei für eine Domain. Unterhalb einer Domain kann sich in der Struktur ein weiterer Teilbaum befinden. Man spricht dann von einer *Subdomain*. Jeder Bereich einer Domain ist durch einen Punkt getrennt und eine Domain ist einzigartig. Abbildung 2.1 zeigt diese Struktur. Der grau hin-

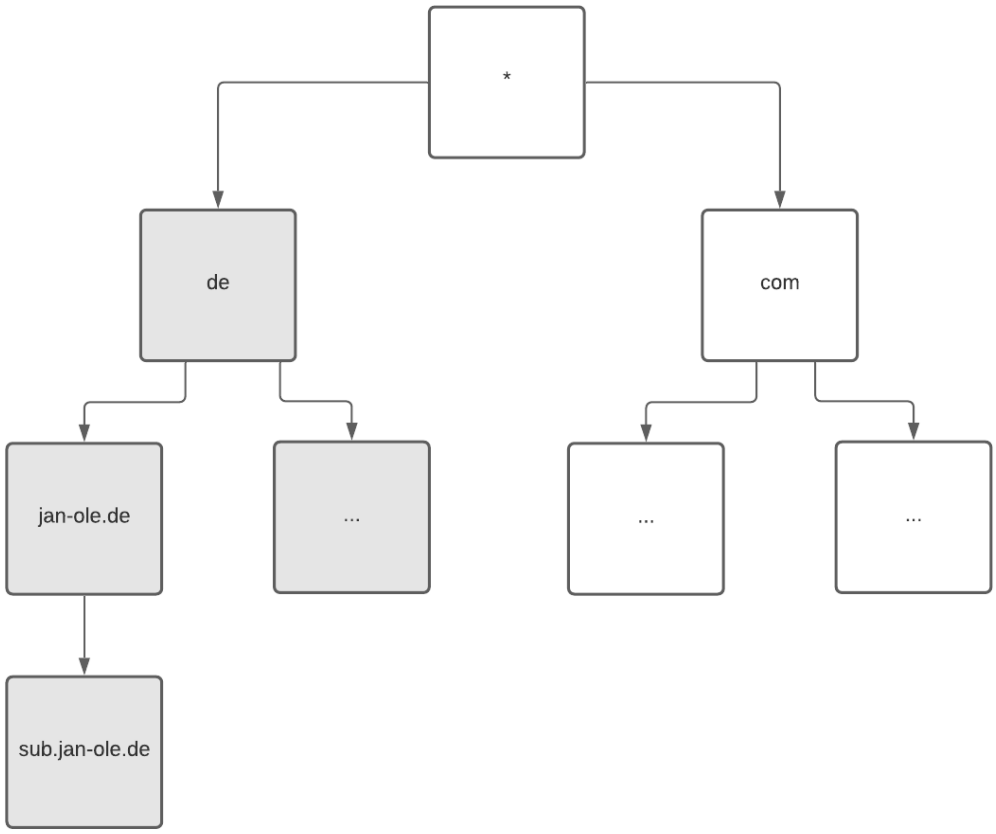

Abbildung 2.1: Baumstruktur des DNS

terlegte Bereich stellt den Bereich der *Top Level Domain* DE da. Dieser wird von der DENIC eG verwaltet. Weiterhin ist jede Domain einzigartig.

2.2 Nameserver und Zonen

Ein Nameserver ist nun im Prinzip ein Datenbankserver dieser verteilten Datenbank. Dieser enthät einen Teil der Daten des Systems und stellt die Daten über den *Resolver* der Außenwelt zur Verfügung. Ein *Resolver* ist also der Teil des Systems, mit dem der Rechner aus Abbildung 1.1 kommuniziert, wenn er eine Anfrage zur *Auflösung* einer Domain stellt. *Auflösen* ist hier also zu verwenden wie 'übersetzen zu IP-Adressen' oder besser das Nachschlagen eines Datensatzes der Datenbank, denn das DNS enthält nicht nur IP-Adressem. Dazu mehr in 2.3.
Eine Zone stellt nun einen Verwaltungsbereich eines oder mehrerer Namespaces dar. Dabei muss eine Zone nicht nicht nur die Daten einer einzigen Domain enthalten.
Ein Nameserver, der eine Zone verwaltet, wird *Authoritative* genannt. Er ist derjenige, der die endgültigen Datensätze enthält. Er muss Anfragen zu seinen Zonen nie weiterleiten. Ein *Authoritative Nameserver* beantwortet außerdem nur Anfragen zu seinen Zonen. Er hält meist keine Daten zu anderen Zonen und wird nur von öffentlichen Nameservern zu seinen Zonen befragt. Andere Anfragen leitet er weiter. Welcher Nameserver für eine Domain als *Authoritative* gilt, ist in einer zentralen Registrierungsstelle verzeichnet. Für die *Top Level Domain* .de ist dies beispielsweise die DENIC eG.

2.3 Records

Die eigentlichen Daten der bereits erwähnten Zonen liegen in sogenannten *Zonen-Dateien*. Diese Dateien sind ähnlich wie die ursprüngliche HOSTS.txt, textbasierte Einträge.

```
$ORIGIN jan-ole.de.
$TTL 86400

@  IN  SOA  jan-ole.de. dns.jan-ole.de. 2021011201 86400 10800 7200 3600

@  IN  NS   ns1.jan-ole.de.
@  IN  NS   ns2.jan-ole.de.
```

Abbildung 2.2: Minimale Zonen-Datei

In 2.3 ist der minimale Aufbau einer Zonen-Datei abgebildet. Einen Zonen-Datei enthält sogennante *Records* bei *Records* handelt es sich im Prinzip um Datensätze der Zone. Der Grundlegende Aufbau eines *Records* enthält folgende Datenfelder:

Name: - Der Namme der betreffenden Domain. Ein Beispiel wäre hier der Wert 'subdomain.jan-ole.de.' Hier ist die Trennung der einzelnen Domains mittel Punkt. So kommt auch der Punkt am Ende zustande. Hiermiet wird auf den Root-Knoten referenziert. Siehe hierzu Abbildung 2.1. Sonderfälle für dieses Feld stellen die werte '@' und '*' dar. @ referenziert die in ORIGIN definierte Zone. Es wird also verwendet wenn es nicht um eine Subdomain geht. Bei '*' handelt es sich um eine Wildcard. Es erhalten also sämtliche Anfragen die selbe Antwort egal welche subdomain angefragt wird.
Time To Live: Wert in Sekunden, der die Gültigkeitsdauer des Resource Records beschreibt. (optional)
Klasse: Protokollgruppe, zu der der Resource Record gehört (optional) Verwendet wird hier fast nur noch die Klasse 'IN' für Internet. Die Anderen Klassen werden deshalb hier nicht betrachtet.
Typ: beschreibt den Typ des Resource Records. Diese Typen werden jeweils in eigenen RFCs spezifiziert und können nachträglich hinzugefügt werden, daher existieren sehr viele dieser Typen. Die Gängigsten Typen sind:

Record Typ	Verwendung
A	Auflösung von IPv4-Adressen
AAAA	Auflösung von IPv6-Adressen
CNAME	Verweis auf weiteren Namen. Kann verwendet werden um einen Alias anzulegen.
MX	Mail Server
NS	Verweis auf weitere Nameserver
PTR	Pointer - Ordnen einer IP-Adresse einen Hostname zu.
TXT	Text
SOA	Start Of Authority - Legt den *Authoritative Nameserver* fest.

Tabelle 2.1: Die Gängigstens DNS-Record Typen

Länge der Daten. Länge der Daten, des Resource Records. Dies ist ein optionales und wird selten verwendet.
Daten: Die eigentlichen Daten des Records. Hier steht Beispielsweise bei einem Record des Typs 'A' die IPv4 Adresse.

Wie in der minimalen Zonen-Datei 2.3 bereits zu erkennen, sind ein SOA-Record und mindestens ein - meistens jedoch zwei - NS-Records Pflicht, damit die Zonen-Datei gültig ist. Der SOA/Start Of Authority-Record enthält den jeweiligen *Authoritative Nameserver*. Ohne diesen Record könnten Daten in vorhandenen Caches nicht Korrekt erneuert werden. Da der Verweis auf den zuständigen DNS-Server fehlt. Auch die Frage, wer für die Verwaltung der Domain zuständig ist, bliebe mit fehlen dieses Records offen.
Die Anzahl der notwendigen NS-Records ist Abhängig von den Vorgaben der einzeilnen Registrierungsstellen. Die DENIC eG schreibt an dieser Stelle das Vorhandensein von min-

destens zwei Records - und damit zwei verschiedenen Nameservern vor. Dies geschieht aus Gründen der Ausfalltoleranz. Ist der Primäre Nameserver z.B. aufgrund von Wartungsarbeiten nicht verfügbar, muss sichergestellt sein, dass Anfragen während dieser Zeit weiterhin beantwortet werden können. Andernfalls würde jede Nichterreichbarkeit des Nameservers dazu führen, dass sämtliche Zonen, die dieser Server verwaltet, nicht Aufgelöst werden können und ebenfalls nicht erreibar sind, da sie keine Anfragen mehr erhalten.
Der TXT-Record stellt ebenfalls einen besonderen Record-Typen dar, da er aufgrund der sehr allgemeinen Inhaltsvorgabe 'Text' vielseitig verwendet werden kann und auch wird. Bekannte Einsatzmöglichkeiten sind z.B. das Ausstellen von SSL-Wildcard-Zertifikaten, oder die Definition von Regeln zu Spam-Prävention.
Der Dienst *LetsEncrypt* bietet TXT-Records als Möglichkeit an um Wildcard-SSL-Zertifikate auszustellen. Für diese muss die Inhaberschaft einer Domain verifiziert werden. Dies geschieht indem der Antragsteller eine Zeichenkette erhält, die er in einen TXT-Record mit dem Namen '_acme-challenge.' schreibt. [Gro21] *LetsEncrypt* stellt daraufhin eine Anfrage für diesen Record, an den für die Domain zuständigen Nameserver. Wenn dieser mit der Zeichenkette antwortet, ist bewiesen, dass der Antragsteller Zugriff auf den *Authoritative Nameserver* der Domain hat. Somit darf er diese verwalten und ein Wildcard-SSL-Zertifikat kann bedenkenlos ausgestellt werden.
Spam-Prävention wird unter Anderem über das sogenannte SPF-Protokoll (Sender Policy Framework) betrieben. Hierfür kann ein TXT-Record angelegt werden der mit 'v=spf1' beginnt und definiert welche Server berechtigt sind, Mails im Namen dieser Domain zu verwenden. Das SPF-Protokoll ist so weit verbreitet, dass es mittlerweile einen eigenen Record Typen 'SPF' gibt. [Kit14] Die Nutzung eines TXT-Records funktioniert aber weiterhin.

2.4 Caching und Delegation

Das Hauptziel von DNS bleibt jedoch die Dezentralisierung der Daten, um Verwaltung zu erleichtern und die Arbeitslast zu verteilen. Dies wird zum einen durch *Caching* und zum anderen durch *Delegation* erreicht. Um die Anzahl der Anfragen an die *Authoritative Nameserver* möglichst gering zu halten und deren Arbeitslast in einer Übersichtlichen Größenordnung zu halten, Gibt es im DNS-Netz öffentliche Nameserver, die darauf ausgelegt sind, große Mengen Gleichzeitiger Anfragen beantworten zu können. Ein Bespiel hierfür ist der öffentliche Nameserver des Anbieters *Cloudflare* mit der IP-Adresse 1.1.1.1
Dieser verwaltet keine Zone selbst, er dient lediglich zur Verteilung zwischengespeicherter Daten. Er stellt Anfragen an die jeweiligen *Authoritative Nameserver* und speichert deren Antworten für eine bestimmte Zeit zwischen. Hier kommt das bereits erwähnte Datenfeld der Time To Live ins spiel. Es gibt den Nameservern genau an dieser Stelle einen Hinweis für welchen Zeitraum der Datensatz zu Speichern ist, bis eine erneute Anfrage gestellt

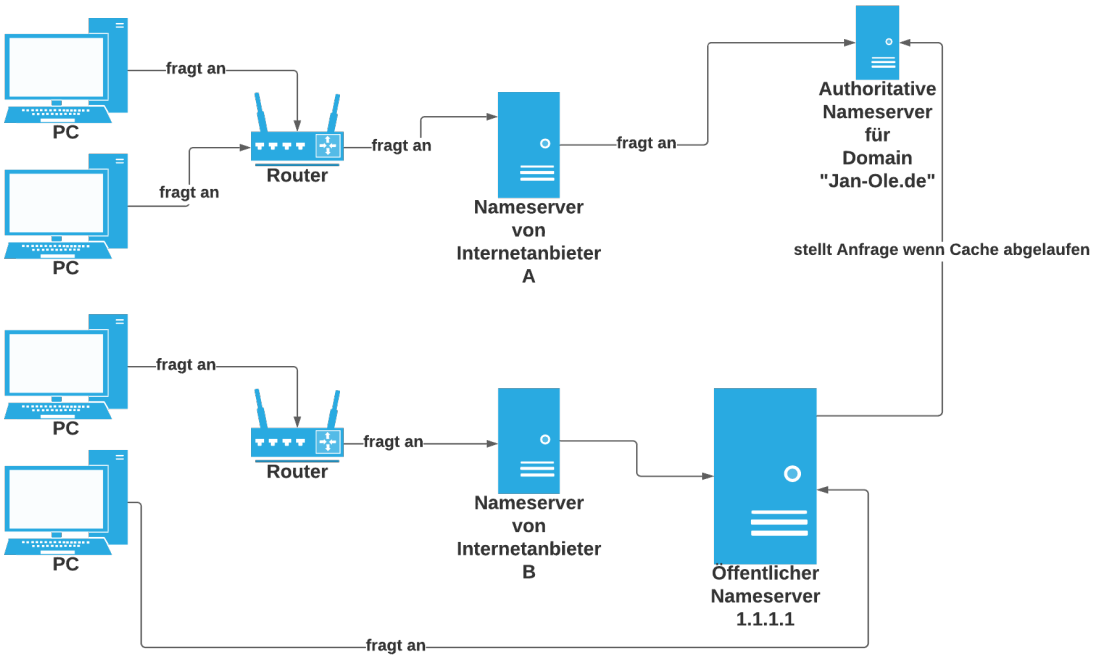

Abbildung 2.3: Die Verteilung von Anfragen

werden soll. In Abbildung 2.3 sind exemplarisch einige Nameserver dargestellt. Die Größe der Server in der Abbildung soll deren Kapazität verdeutlichen. je größer der Server, desto größer die Kapazität in Hinblick auf Beantwortung gleichzeitiger Anfragen. Aus Gründen der Übersichtlichkeit sind nur die Anfragen dargestellt. Antworten erfolgen in der Abbildung immer vom selben Server, der die Anfrage auch erhalten hat.
Hingewiesen sei darauf, dass ein Client seine Nameserver selbst wählen kann. Zu sehen ist dies zum einen an dem PC, dessen Anfrage nicht über seinen Router läuft, zum anderen am Server des Internetanbieters B, der seine Anfrage an Cloudflare stellt, statt direkt den zuständigen Nameserver zu fragen. Ebenfalls sei angemerkt, dass hier von Routern für den Heimanwenderbereich gesprochen wird. Router die in einem Professionellen Umfeld verwendet werden, haben meist, keinen eigenen DNS Cache.
Stellt ein PC nun einer Anfrage an seinen Router, schaut dieser in seinen Cache und beantwortet die Anfrage gegebenenfalls direkt mit dem gecachten Wert. Nur Wenn dieser nicht Vorhanden ist, leitet er die Anfrage an einen öffentlichen Nameserver weiter. Konfiguriert man diesen nicht selbst, wird ein Nameserver verwendet, den der Internetanbieter zur Verfügung stellt. Nun schaut auch dieser Server in seinen Cache und beantwortet gegebenenfalls die Anfrage. Dieses Vorgehen kann in der Theorie nun beliebig häufig wiederholt werden, bis entweder ein gecachter Wert im Netz gefunden wird, oder der *Authoritative Nameserver* befragt wurde.
Die Kette der Anfragen wird aber nach Möglichkeit sehr kurz gehalten, um die Antwortzei-

ten auf die Anfrage des Benutzers nicht zu erhöhen.
Wurde nun ein Datensatz gefunden mit dem die Anfrage beantwortet werden kann, durchläuft dieser die beteiligten Server rückwärts und die Server erneuern ihren Cache.
Hierbei kommt es zu einem Dilemma die bereits erwähnte *Time To Live* ist ein Timer für den Ablauf der Gültigkeit des Eintrags, dieser wird in der Antwortkette jedes mal neu gestartet, wenn die Anwort eines Servers erhalten wird. Konkret bedeutet das: Die *Time To Live* gilt für jeden Schritt der Anfrage. Haben wir beispielsweise einen Eintrag mit einer Gültigkeit von 3600 Sekunden - also einer Stunde und der Router eines Privatanschlusses stellt eine Anfrage an den öffentlichen Nameserver von *Cloudflare* und dieser antwortet. Dann kann es passieren, dass dies eine Sekunde vor Ablauf der Gültigkeit geschieht, der Router erhält also eine Antwort, die unter Umständen eine Sekunde später veraltet ist, startet aber trotzdem in seinem Cache einen neuen Timer von 3600 Sekunden. Dies bedeutet, dass die *Time To*

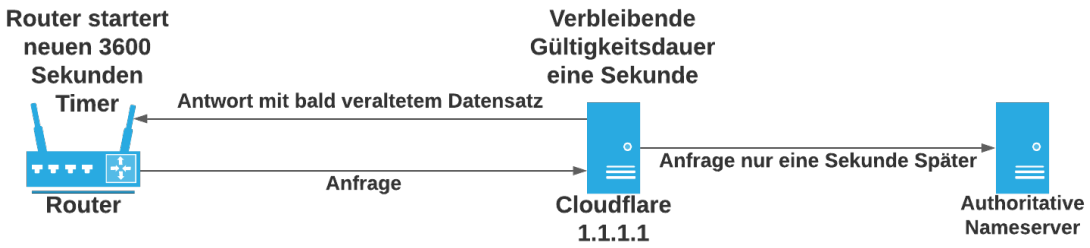

Abbildung 2.4: Problem mit veralteten Einträgen bei zu hoher TTL

Live weder zu kurz noch zu lang gewählt werden sollte. Ist sie zu lang, dauert die Verteilung aktueller Datensätze sehr lang und die Benutzer erhalten eventuell veraltete Daten. Wählt man die *Time To Live* hingegen zu kurz, laufen Caches sehr schnell ab, was dazu führt das der oder die *Authoritative Nameserver* sehr viele Anfragen erhalten, unter hoher Last stehen und der Effekt der Datenverteilung zunichte gemacht wird. Eine geeignete *Time To Live* hängt demnach von der Frequenz der Änderungen (von beispielsweise IP-Adressen), der Anzahl der Anfragen an die Domain sowie der Leistungsfähigkeit des *Authoritative Nameservers* ab. Hier muss eine gute Balance gefunden werden.

Mit Caching wird also die Verteilung der Arbeitslast unterstützt. *Delegation* hingegen dient der verteilten Verwaltung. Wie in Kapitel 2 (Abbildung 2.1) bereits erwähnt können verschiedene Zonen von verschiedenen Servern verwaltet werden. Jede Subdomain kann also einer anderen Organisation zugewiesen sein. Hierfür muss ledglich der in Tabelle 2.1 bereits gennante SOA-Record zu der jeweiligen Subdomain auf dem zuständigen Server vorhanden sein und der Verwalter der übergeordneten Zone sie zuständigen Nameserver eintragen. Registriert man sich beispielsweise eine .de Top-Level-Domain bei der DENIC eG, teilt man einen gewünschten, zuständigen Nameserver mit und die DENIC eG delegiert daraufhin die

zuständigkeit an diese Server. Dies passiert über einen NS-Record für die jeweiligen Subdomain in der übergeordneten Domain, der dann auf den neun Server zeigt. Gleiches ist für jede weitere Subdomain möglich.

```
kunde1  IN  NS   ns1.kunde1.jan-ole.de.
```

Abbildung 2.5: Delegation mittels NS-Record

Mit dem in Abbildung 2.4 gezeigten Datensatz wird beispielsweise die Zuständigkeit für die Subdomain *kunde1.jan-ole.de* an den Nameserver *ns.kunde1.jan-ole.de* übergeben. Der Eintrag muss lediglich (zusammen mit oben erwähnten SOA-Record für die Subdomain) auf dem Zielserver vorhanden sein.

```
$ORIGIN kunde1.jan-ole.de.
$TTL 86400

@  IN  SOA  kunde1.jan-ole.de. ...
ns1 IN  A   10.11.12.13
```

Abbildung 2.6: Zonendatei auf dem Zielserver

Abbildung 2.4 zeigt einen beispielhaften Eintrag auf dem Server der Subdomain. Der SOA-Record ist dabei aus Gründen der übersichtlichkeit gekürzt. Der Zielserver betrachtet sich daraufhin für die Zone *kunde1.jan-ole.de.* als *Authoritative Nameserver* und die IPv4-Adresse von *ns1.kunde1.jan-ole.de* löst zu 10.11.12.13.

3 Angriffe und Probleme

Wie bei jeder Technologie gibt es auch im DNS Möglichkeiten zum Missbrauch. Durch den flexiblen Aufbau des DNS konnten die meisten bekannten Angriffe zumindest deutlich erschwert werden.
Als Beispiel hierfür soll das sogenannte *DNS Cache Poisining* betrachtet werden. Dabei wird das in Abschnitt 2.4 erläuterte Caching ausgenutzt, um die Anfragen eines Opfers auf andere Server umzuleiten, ohne dass dieses die Umleitung bemerkt. Betrachten wird dazu erneut die Abbildung 2.4. Sendet der dortige Router eine Anfrage an den DNS-Server, so erhält diese Anfrage eine ID - beispielsweiser 100.
Alles was nun passieren müsste, um eine falsche Auflösung der Adresse zu erzeugen, ist eine Antwort mit der passenden ID zu senden, bevor die eigentliche Antwort des gefragten Servers zurück kommt. Das ist möglich, da durch die vielen Netzwerk-Hops einer rekursiven Anfrage ein relativ großes Zeitfenster entsteht in dem eine solche Antwort gesendet werden kann.
Ist dies erfolgreich gewesen, so legt der angegriffene Server den flschen Eintrag in seinen Cache. Das Problem das dadurch entsteht ist aber noch viel größer, denn der Server liefert den falschen Eintrag für die Dauer der *Time To Live* an alle anfragenden Server aus. Die widerrum speichern den Eintrag selbst für die Dauer der *Time To Live*. Diese kann vom ANgreifer ebenfalls auf einen sehr hohen Wert gesetzt werden. So kann sich ein gefälschter Eintrag sehr lange im Netz halten.
Wie aber findet man nun die passende ID? In der frühen Zeit von DNS wurde diese einfach inkrementiert. Der angreifende Server stellt also eine Anfrage zum Auflösen einer Domain für die er tatsächlich zuständig ist und kennt somit die aktuelle ID. Daraufhin muss er nur noch viele gefälschte Pakete mit größeren IDs senden. Die wahrscheinlichkeit für einen Erfolg ist so sehr hoch.
Nachdem dies zu mehreren großen Problemen führte, wurde unter anderem die Vergabe einer Anfrage-ID verändert. Diese wurde daraufhin zusammen mit den Quell-Port der Anfrage zufällig vergeben. So muss nicht nur eine 16 Bit-Zahl für die ID erraten werden, sondern zusätzlich der Quellport der Anfrage. Dies sorgt dafür dass ein solcher Angriff sehr viel schwieriger wird. Das Problem jedoch besteht weiterhin.

4 Ausblick

4.1 DNS over TLS und DNSSEC

Zur Behebung des angesprochenen Problems der Anfragenmanipulation wurde *DNSSEC*. *DNSSEC* bietet eine Möglichkeit, Anfragen mittels Public/Private-Key Signaturen zu verifizieren. Dafür bietet DNS verschieden Record-Typen, um Informationen für die Signierung von Anfragen zur Verfügung zu stellen. Beispielhaft sei hier der DS-Record (Delegation of Signing) genannt.
Auf das genaue Verfahren soll im Rahmen dieser Arbeit nicht eingegangen werden.
DNSSEC ist aufgrund hoher Komplexität noch nicht weit verbreitet. In der Zukunft wird *DNSSEC* - ähnlich wie HTTPS - aber zur Pflicht werden, da damit die Manipulation von Anfragen so gut wie unmöglich wird.
Eine weitere Technologie im zusammenhang mit DNS ist *DNS over TLS* und *DNS over HTTPS*. Dies dient zur Verschlüsselung der DNS-Anfragen. Diese sind nach aktuellem Standard noch meist unverschlüsselt und können deshalb mitgelesen werden. Dies ermöglicht eine Erkennung der Server zu denen der Benutzer eine Verbindung aufbaut. Mit der wachsenden Sorge um Privatsphäre gewinnt auch die Verschlüsselung dieser Anfragen an Wichtigkeit. *DNS over HTTPS* sendet dazu normale HTTPS-Anfragen an einen Webserver. Dieser übernimmt dann das Senden der Antwort. Das erschwert die Erkennung von DNS-Anfragen, da HTTPS und DNS nicht mehr unterschieden werden können. *DNS over HTTPS* ist allerdings noch kaum in Verwendung.

4.2 Zusammenfassung

Bei dem *Domain Name System* handelt es sich um ein solides System, dass die meisten Nutzer nicht einmal bemerken. Wie in dieser Arbeit erwähnt hat auch das DNS durchaus Schwächen, kann aber stets flexibel reagieren. Beispielhaft hierfür wurde der Umgang mit geänderte Anforderungen wie z.B. die Verschlüsselung von Anfragen erläutert. Das größte Problem bleibt hier die Verbreitung der angebotenen Lösungen. Weiterhin bietet das DNS oft zusätzlich die Grundlage für für unterstützende Funktionen anderer Protokolle oder Anwendungen, wie beispielsweise die Filterung von Spam. Mit Funktionen wie *DNS over TLS* oder der - in dieser Arbeit weniger betrachteten - Unterstützung von IPv6 ist die Kompatibilität mit modernen, zukünftigen Funktionen des Internets sichergestellt.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[eG21] DENIC eG. *DENIC Statistiken*. 6. Jan. 2021. URL: https://www.denic.de/wissen/statistiken/.

[Gro21] Internet Security Research Group. *Challenge Typen*. 13. Jan. 2021. URL: https://letsencrypt.org/de/docs/challenge-types/.

[Kit14] S. Kitterman. *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. Apr. 2014. URL: https://tools.ietf.org/html/rfc7208.

[LA06] C. Liu und P. Albitz. *DNS and BIND: Help for System Administrators*. O'Reilly Media, 2006. ISBN: 9780596553401. URL: https://books.google.de/books?id=HggtWI1ShvMC.

[Moc83] Paul Mockapetris. *DOMAIN NAMES - CONCEPTS and FACILITIES*. Nov. 1983. URL: https://tools.ietf.org/html/rfc882.

[Pos84] J. Postel J.;Reynolds. *Domain Requirements*. Okt. 1984. URL: https://tools.ietf.org/html/rfc920.