Telematics
Computer Systems

Freie Universität Berlin

Bachelorarbeit

Schutz wichtiger Webseiten durch RPKI

Messung und Analyse

Robert Schmidt

Matr. 4515102

Betreuer: Prof. Dr.-Ing. Jochen Schiller
Dipl.-Inf. Matthias Wählisch

Institut für Informatik, Freie Universität Berlin, Deutschland

19. Januar 2014

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, sind als solche gekennzeichnet. Die Zeichnungen oder Abbildungen sind von mir selbst erstellt worden oder mit entsprechenden Quellennachweisen versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner Prüfungsbehörde eingereicht worden.

Berlin, den 19. Januar 2014

(Robert Schmidt)

Kurzzusammenfassung

Webdienste gehören zu den wichtigsten Anwendungen des Internets. Ihr Funktionieren hängt maßgeblich von einem fehlerfreien Routing ab. Das heutige Inter-Domain Routing wird von BGP bestimmt, welches in seiner ursprünglichen Definition keinen Schutz vor BGP-Hijacking bietet. Seit Januar 2012 können Besitzer von IP-Präfixen mit Hilfe des RPKI-Standards die Bekanntgabe von Präfixen durch Autonome Systeme kryptographisch absichern. Diese Arbeit untersucht den Schutz wichtiger Webseiten gegen BGP-Hijacking. Dazu werden die Webserver-IP-Adressen im BGP-Routing untersucht, inwieweit dafür Resource Public Key Infrastructure (RPKI) eingesetzt wird.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Einleitung

Das Internet, entstanden durch den Zusammenschluss einzelner Computer und Rechenzentren, hat sich innerhalb kürzester Zeit zu einem weltweit verfügbaren Netzwerk entwickelt und ist heute das wichtigste Medium zum Austausch von Informationen. Ob beim Militär, an Universitäten, in der Wirtschaft oder Politik, überall in der Gesellschaft wird das Internet benutzt und ist aus vielen Bereichen kaum noch wegzudenken. Ausfälle oder auch nur Beeinträchtigungen können schwerwiegende Folgen haben und wären überall zu spüren.

Zum Austausch der Informationen werden Daten im Internet in Paketen über ein Netzwerk von Routern verschickt. Sie entscheiden mit Hilfe des Internet Protokolls (IP) für jedes Paket über welche Verbindung die Daten verschickt werden. Dieser Prozess wird Routing genannt. Router tauschen dazu selbst Informationen über Routing-Protokolle aus, um so dezentral die bestmögliche Verbindung auswählen zu können. Das wichtigste Routing-Protokoll ist das BGP, welches insbesondere zwischen Teilnetzwerken des Internets, den Autonomen Systemen (AS), benutzt wird. BGP beruht ursprünglich auf einer Vertrauensbasis zwischen den Routern und weist damit Schwachstellen auf. Ein Router kann die über BGP verteilten Informationen nicht zuverlässig überprüfen. Dadurch kann das Routing bewusst manipuliert bzw. durch Fehlkonfiguration unbewusst verändert werden [10].

Um das Inter-Domain Routing vor Fehlern zu schützen, wurde die Resource Public Key Infrastructure (RPKI) einschließlich Verifikationsmechanismen von Routing-Informationen entwickelt. Einige Schwachstellen können durch Authentifizierung vermieden werden. Aber dennoch können IP-Präfixe entführt werden (BGP-Hijacking). Die RPKI erlaubt die kryptografisch gesicherte Bestätigung des Besitzes eines IP-Präfixes, indem die Informationen der IP-Adressvergabe in einer PKI hinterlegt werden. BGP-Router können diese Informationen nutzen und das Verteilen falscher IP-Präfixe unterbinden. RPKI muss durch die Betreiber der Autonomen Systeme eingesetzt werden. Aktuell sind allerdings nur ca. 4% aller IP-Präfixe geschützt [1].

Webdienste gehören zu den wichtigsten Anwendungen des Internets. Ihr Funktionieren hängt maßgeblich von einem fehlerfreien Routing ab. Deshalb wird in diese Arbeit anhand wichtiger Webseiten untersucht, inwieweit diese gegen BGP-Hijacking geschützt sind bzw. ob die Webserver-IP-Adressen im BGP-Routing durch RPKI gesichert sind.

Große wichtige Webseiten benutzen meist mehrere Webserver oder verwenden Content Delivery Networks (CDN), um ihre Inhalte schnell dem Endkunden bereitzustellen. Diese sind über mehrere Netzwerkstandorte verteilt. Dieser Aspekt muss bei der Untersuchung berücksichtigt

[1] RPKI-Dashboard: http://rpki.surfnet.nl

werden, damit alle Webserver zu einer Webdomain bei der Messung beachtet werden können.

In einer Einführung wird die Verteilung von Webinhalten, anschließend das Prinzip des Routings und das BGP beschrieben, sowie das Problem BGP-Hijacking mit der Lösung RPKI. Im nächsten Kapitel werden die einzelnen Schritte der Untersuchungsmethode erläutert. Dazu werden mittels ORDNS-Servern an verschiedenen Netzwerkstandorten DNS-Anfragen für alle wichtigen Webdomains gestellt, den so ermittelten IP-Adressen werden IP-Präfix und ASN aus BGP-Routing-Tabellen zugeordnet und dafür je der RPKI-Status bestimmt. Das nachfolgende Kapitel beschreibt deren Implementierung. Dabei werden mit Hilfe von Perl-Skripten parallelisiert DNS-Abfragen gestellt und ein IP-Trie für die Zuordnung IP-Adresse zu IP-Präfix und ASN selbst entwickeltet. Im anschließenden Kapitel wird eine ORDNS-Server-Liste zusammengestellt. Nach der Durchführung der Untersuchung werden die Ergebnisse analysiert und ausgewertet. Das Endergebnis folgt mit einem Ausblick in der abschließenden Zusammenfassung.

2 Hintergrund

2.1 Verteilung von Webinhalte

Große Webseiten werden über mehrere Webserver angeboten. Dazu wird zusätzlich meist ein CDN verwendet, um die Inhalte für den Endbenutzer optimiert schnellen bereitzustellen. Beim Aufruf einer Webseite wird über DNS die IP-Adresse des Webservers ausgewählt, von dem die Webseite geladen werden kann.

2.1.1 Domain Name System

Das Domain Name System (DNS) [29], [30] dient zur Auflösung von Namen und Diensten von Computern in einem Netzwerk. Es wird benutzt um Namen IP-Adressen zu zuordnen.

Für die Verwaltung von Namen wird eine Baumstruktur (DNS-Hierarchie) verwendet. Jeder Knoten hat ein Label (Name), der ihn eindeutig relativ zum Elternknoten identifiziert. Die Wurzel (root) hat verschiedene Kinderknoten, die Top-Level-Domains (TLD), die wiederum Teilbäume bilden (Namensraum). Ein Domain-Name setzt sich aus den einzelnen Labels ausgehend vom Blatt bis zur Wurzel zusammen, wobei die Labels durch „.“ Punkt getrennt werden. Durch diese Struktur sind die Namen eindeutig und die Verantwortung der einzelnen Teilbäume kann aufgeteilt werden.

Es gibt DNS-Server, die für je einen unterschiedlichen Namensraum zuständig sind und DNS-Anfragen zur Namensauflösung beantworten.

Der Client (DNS-Resolver) fängt bei der Namensauflösung beim ersten Namensraum an, den untersten Knoten im Baum, und fragt den zuständigen DNS-Server. Wenn der Client den DNS-Server für den ersten Namensraum noch nicht kennt, fragt er den DNS-Server für den zweiten Namensraum. Wenn dieser auch unbekannt ist, wird der nächste gefragt, solange bis die Root-DNS-Server gefragt werden, die bekannt sind. Der Root-Server antwortet mit der IP-Adresse des zuständigen DNS-Servers des TLD-Namensraums. Der kann wiederum gefragt werden für den DNS-Server des nächsten Namensraums, solange bis der gesuchte Name aufgelöst werden kann.

Über die Domain-Namen können verschiedene Informationen abgefragt werden. Für jeden Typ gibt es einen Eintrag (DNS-Record) nach dem gefragt werden kann:

- A für IPv4 Eintrag
- AAAA für IPv6 Eintrag

- PTR für Hostname
- MX für Mailserver
- CNAME für Alias
- NS für Nameserver, der für diesen Bereich zuständig ist.

Es gibt rekursive und iterative Abfragen. Bei einer iterativen Anfrage stellt der Client alle DNS-Anfragen und fragt so jeden Server bis er den gesuchten Record gefunden hat. Der Client fragt seinen DNS-Server und bekommt entweder den gesuchten Record oder ein oder mehrere Verweise auf andere DNS-Server. Der Verweis enthält den Namen des Servers und ggf. die IP-Adresse. Der Client wählt einen DNS-Server aus den Verweisen aus und fragt ihn. Dieser antwortet mit dem gesuchten Record oder mit weiteren Verweisen, usw.

Bei der rekursiven Anfrage stellt der gefragte DNS-Server die iterativen DNS-Anfragen im Auftrag des Clients. In Abbildung 2.1 wird der Verlauf einer rekursiven DNS-Abfrage gezeigt. Dem DNS-Server „8.8.8.8“ wird eine rekursive DNS-Anfrage für „maps.google.de“ gestellt.

```
C:\>dig @8.8.8.8 +trace maps.google.de

; <<>> DiG 9.6-ESV-R9-P1 <<>> @8.8.8.8 +trace maps.google.de
; (1 server found)
;; global options: +cmd
.                       15504   IN      NS      a.root-servers.net.
.                       15504   IN      NS      b.root-servers.net.
.                       15504   IN      NS      c.root-servers.net.
.                       15504   IN      NS      d.root-servers.net.
.                       15504   IN      NS      e.root-servers.net.
.                       15504   IN      NS      f.root-servers.net.
.                       15504   IN      NS      g.root-servers.net.
.                       15504   IN      NS      h.root-servers.net.
.                       15504   IN      NS      i.root-servers.net.
.                       15504   IN      NS      j.root-servers.net.
.                       15504   IN      NS      k.root-servers.net.
.                       15504   IN      NS      l.root-servers.net.
.                       15504   IN      NS      m.root-servers.net.
;; Received 228 bytes from 8.8.8.8#53(8.8.8.8) in 41 ms

de.                     172800  IN      NS      z.nic.de.
de.                     172800  IN      NS      n.de.net.
de.                     172800  IN      NS      l.de.net.
de.                     172800  IN      NS      f.nic.de.
de.                     172800  IN      NS      a.nic.de.
de.                     172800  IN      NS      s.de.net.
;; Received 346 bytes from 199.7.91.13#53(d.root-servers.net) in 199 ms

google.de.              86400   IN      NS      ns1.google.com.
google.de.              86400   IN      NS      ns2.google.com.
google.de.              86400   IN      NS      ns3.google.com.
google.de.              86400   IN      NS      ns4.google.com.
;; Received 114 bytes from 195.243.137.26#53(s.de.net) in 36 ms

maps.google.de.         345600  IN      CNAME   maps.l.google.com.
maps.l.google.com.      300     IN      A       173.194.69.100
maps.l.google.com.      300     IN      A       173.194.69.138
maps.l.google.com.      300     IN      A       173.194.69.139
maps.l.google.com.      300     IN      A       173.194.69.113
maps.l.google.com.      300     IN      A       173.194.69.101
maps.l.google.com.      300     IN      A       173.194.69.102
;; Received 159 bytes from 216.239.36.10#53(ns3.google.com) in 35 ms
```

Abb. 2.1: rekursive DNS-Abfrage für maps.google.de

Dieser fragt zuerst einen der Root-Server (d.root-servres.net) nach „.de“. Von ihm bekommt er Verweise auf mehrere „de“-Nameserver. Davon wird „s.de.net“ nach „google.de“ gefragt. Dieser antwortet mit Nameservern von Google. Davon wird „ns3.google.com“ ausgewählt und nach „maps.google.de“ gefragt. Dieser antwortet mit einem CNAME auf „maps.l.google.com“. Es müsste jetzt diese Domain aufgelöst werden. Aber da der gleiche Nameserver dafür zuständig ist, werden die IP-Adressen für „maps.l.google.com“ gleich in der Antwort mitgeliefert.

Bei der rekursiven Abfrage verschiebt sich die Last vom Client auf den Server.

Die meisten DNS-Server reagieren nur auf rekursiven Anfragen von Clients aus ihrem Netzwerk, meistens nur der lokale DNS-Server.

Es gibt aber öffentliche DNS-Server die rekursive Anfragen bearbeiten, die von fremden bzw. beliebigen Clients kommen können. Das sind Open Recursive DNS-Server (ORDNS).

2.1.2 Content Delivery Network

Die meisten IP-Pakete im Internet bringen Inhalte wie Webseiten (52%), Bilder und Videos (3%), zum Endbenutzer [20]. Damit das im Sinne des Endkunden so schnell wie möglich geht, gibt es Content Delivery Network (CDN). Das ist ein Netzwerk von Servern verteilt über das Internet, die die Inhalte für Endbenutzer optimiert, schnell bereitstellen.

Internet-Inhalteanbieter, wie Webseitenbetreiber, benutzen für ihre Inhalte CDNs. Sie können selbst ein CDN betreiben oder sie benutzen einen CDN-Betreiber, der diese Funktion als Dienstleistung anbietet, und sparen dadurch die nötige Infrastruktur. Große CDNs sind zum Beispiel Akamai, LimeLight oder Amazon Web Services.

Das CDN bietet zum einen Server-Ressourcen („Cloud“) zum Speichern der Inhalte und zum anderen ein verteiltes Caching sowie ein Routing-System, das Client-Anfragen auf entsprechende Server umlenkt. Die Server werden so im Internet platziert, dass Endbenutzer die Inhalte schnell laden können.

Ein CDN besteht aus:

- **Replikationsserver** (Replica Server) speichern die Kopien der Inhalte; meist nur der Inhalte, die für die Endbenutzer in der Nähe interessant sind bzw. sein könnten.
- **Verteilungssystem** (Distribution System) kopiert bzw. verteilt die Inhalte auf Replikationsserver.
- **Request-Routing-System** leitet Nutzeranfragen auf einen optimalen Replikationsserver um. Für die Bewertung werden verschiedene Parameter benutzt.
- **Accouting-System** führt Statistiken über die Nutzung und Auslastung der Server und andere Parameter, die vom Routing-System benutzt werden.

Das Request-Routing-System wählt den optimalen Replikationsserver aus, indem die Entfernung zwischen Client und Server bewertet wird. Dazu wird zum Beispiel anhand der anfragenden Client-IP-Adresse die geographische Position mit der des Servers verglichen oder der Abstand wird anhand der Anzahl der Zwischenstationen (Hops) über die TTL (Time-To-Live) bestimmt. Aber auch Daten vom Accouting-System wie Serverauslastung und Netzwerklast des Replikationsservers fließen mit ein.

Es gibt verschiedene Techniken die Client-Anfrage umzuleiten: zum Beispiel DNS-Namensauf-

lösung, HTTP-Redirection, IP-Anycast oder Peer-to-Peer-Routing [33]. Meistens wird dazu DNS verwendet.

Der Inhalteanbieter veröffentlicht den Inhalt über die CDN-Server und erhält vom CDN ein URL-Pfad bestehend aus einer DNS-Subdomain vom CDN und einem Pfad zum Inhalt. Die URL identifiziert den Inhalt. Der Inhalteanbieter kann zum einen den Inhalt in seiner Webseite über die URL vom CDN referenzieren (embedded Content) oder eine Subdomain für mehrere Inhalte benutzen und über ein DNS-CNAME Anfragen dazu auf den DNS-Server des CDNs leiten.

Beim Laden der Webseite wird über den URL-Verweis der Inhalt vom CDN geladen. Dabei wird die Domain der URL mittels DNS aufgelöst und somit der DNS-Server des CDNs gefragt. Der CDN-DNS-Server gibt die IP-Adresse des Replikationsservers zurück, der vom Request-Routing-System ausgewählt wurde. Das Routing-System benutzt dabei die Quell-IP-Adresse der DNS-Anfrage, normalerweise die Client-IP-Adresse [15], [16], [39], [40].

Die Abbildung 2.2 zeigt ein Beispiel für den Ablauf einer Client-Anfrage. Der Client lädt die Webseite von „domain.com“ mit einem Bild-Verweis auf „234.cdn.com/567image.jpg“. Um das Bild herunterladen zu können, löst er die Domain „234.cdn.com“ auf. Dafür fragt der Client nach „234.cdn.com“ den DNS-Server des CDNs „cdn.com“ und erhält über das Routing-System des CDNs eine IP-Adresse des optimalen Replikationsservers. Von diesem lädt der Client das Bild „567image.jpg“.

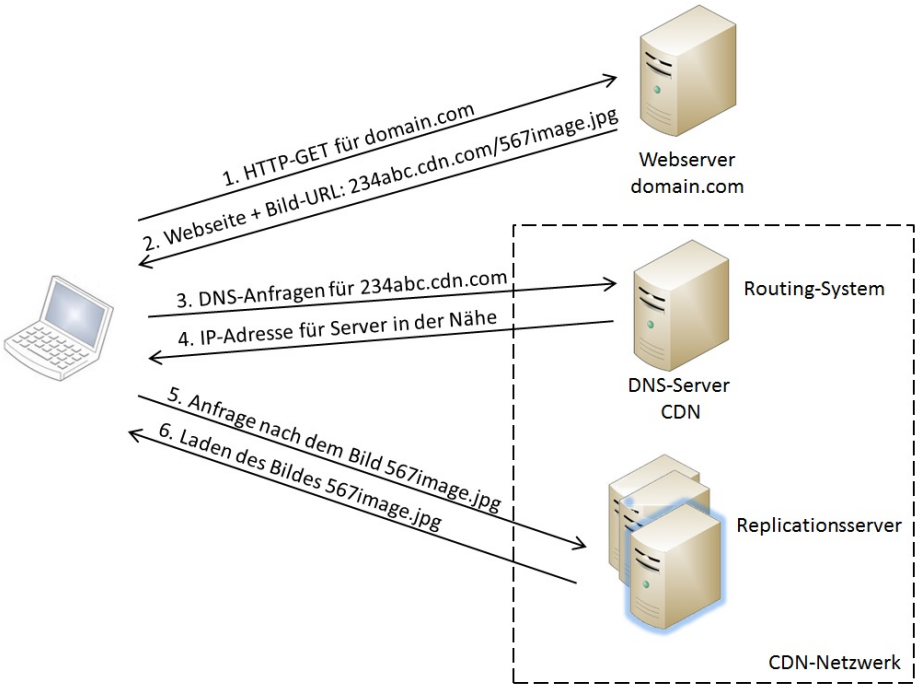


Abb. 2.2: Client-Anfrage bei einem CDN, das DNS benutzt

2.1.3 Übersicht verschiedener CDN-Messmethoden

CDNs sind in der Regel geschlossene Systeme. Das heißt es ist nicht genau bekannt, wie sie intern funktionieren. Deshalb wurden verschiedene Messungen durchgeführt, bei denen versucht wurde, die interne Struktur und Funktionsweise zu ermitteln (Reverse Engineering). Da ein CDN-Netzwerk über das Internet verteilt ist, werden Messungen dazu an verschiedenen Netzwerkstandorten durchgeführt. Es gibt dabei verschiedene Ansätze:

- **ORDNS-Server**
  Eine Möglichkeit besteht darin ORDNS-Server zu benutzen. Dazu werden DNS-Anfragen an ORDNS-Server gestellt, die sie rekursiv auflösen und damit den CDN-DNS-Server anfragen. Dieser wählt anhand der anfragenden Client-IP-Adresse, in diesem Fall der ORDNS-Server, den Replikationsserver aus.

  ORDNS-Server können direkt aus bestehenden Listen benutzt werden oder es wird eine neue Liste zusammengestellt. Diese Liste kann zum einen aufgestellt werden, indem von einer Webseite Client-IP-Adressen gesammelt werden und dann über reverse-DNS-Lookup der zuständige DNS-Server der Client-IP-Adresse ermittelt wird, um ihn dann auf ORDNS zu testen [15], [16]. Eine andere Möglichkeit ist es, DNS-Server aus einer Log-Datei von einem großen DNS-Server zu sammeln und diese auf ORDNS zu testen oder es werden DNS-Server von großen wichtigen Webseiten getestet [24]. DNS-Server können auch über embedded Content gesammelt werden. Dabei wird zum Beispiel ein Bild auf mehreren Webseiten eingebettet. Die Bild-URL verweist auf einen Server der Forscher. Ein Client lädt die Webseite und dabei auch das Bild vom anderen Server. Dabei wird die URL-Domain über DNS aufgelöst und der DNS-Server der Forscher angefragt, der dann alle IP-Adressen aus DNS-Abfragen speichert [26].

- **PlanetLab-Netzwerk**
  Außerdem kann auch das PlanetLab-Netzwerk[1] benutzt werden. Das ist ein weltweites Forschungsnetzwerk bestehend aus mehreren Knoten, auf denen verschiedene Netzwerkdienste oder Software getestet, gemessen bzw. ausgeführt werden können.

  Das Netzwerk kann verwendet werden, um global verteilt DNS-Server abzufragen. Dabei wird der lokale DNS-Server des jeweiligen PlanetLab-Knotens benutzt [1], [39], [40].

  Es gibt aber auch ein Internet-Mess-Tool DipZoom[2], das im PlanetLab-Netzwerk läuft und zusätzlich auch auf anderen Rechnern ausgeführt werden kann [42]. Dadurch kann eine bessere Verteilung erzielt werden.

- **Messkampagne**
  Eine Messkampagne ist eine weitere Möglichkeit DNS verteilt abzufragen. Dazu wird ein Skript oder Programm auf einer Webseite veröffentlicht und jeder eingeladen das Programm auf seinem Computer lokal auszuführen. Das Programm führt die Messungen bzw. die DNS-Abfragen aus. Dabei wird jeweils der lokale DNS-Server des Benutzers verwendet [2], [3].

[1] https://www.planet-lab.org/
[2] http://dipzoom.case.edu/

- **Bannerwerbung**
  Eine andere Option besteht darin, ein Werbenetzwerk zu benutzen, welches Anzeigen bzw. Banner auf mehreren Webseiten veröffentlicht. Es wird ein iframe mit JavaScript-Code als Werbeanzeige über das Netzwerk verteilt. Der Code lädt für ein paar Test-Domains (in dem Beispiel nur 27 [23]) ein Bild und sendet die Test-Daten an einen Server der Forscher. Der Browser löst dadurch die Domain über seinen lokalen DNS-Server auf [23].

Für die Bewertung der verschiedenen Ansätze übers Internet verteilte Messungen durchzuführen, ist zum einen die geographische und topologische Verteilung der Messpunkte wichtig, zum anderen ist auch die Umsetzung der Messinfrastruktur zu beachten, ob beispielsweise zusätzliche Hardware-Ressourcen gebraucht werden.

Das PlanetLab-Netzwerk ist unzureichend gut verteilt und für eine global verteilte Messung zu ungenau, da Knoten überwiegend in Nordamerika und Europa aber auf den anderen Kontinenten kaum bis gar nicht vorhanden sind. Zusätzlich stehen die Knoten überwiegend in Forschungs- und Universitätsnetzwerken und erzielen so auch keine gute topologische Verteilung. Es können aber Ressourcen gespart werden, da eine bestehende Infrastruktur benutzt werden kann.

Das Internet-Mess-Tool DipZoom läuft momentan auf insgesamt 118 Servern. Deshalb wird hierbei keine bessere Verteilung als bei PlanetLab mit ca. 560 Messpunkte erreicht.

Bei einer Messkampagne sind die teilnehmenden Benutzer(anzahl) als auch die Verteilung der Beobachtungspunkte ungewiss, bzw. eine Messkampagne kann durchaus länger als erwartet dauern bis ausreichend viele Benutzer teilnehmen und damit eine gute Verteilung erreicht ist. Um viele Benutzer anzusprechen bzw. einzuladen ist ein zusätzlicher Aufwand nötig. Trotzdem können hier sonst evtl. nötige Hardware-Ressourcen gespart werden.

ORDNS-Server können benutzt werden, um DNS-Abfragen von unterschiedlichen Beobachtungspunkten des Internets zu stellen. Allerdings ist es schwierig, eine Liste von gut verteilten Servern zu finden. Damit aus einer Log-Datei von einem DNS-Server oder von einer großen Webseite IP-Adressen gesammelt werden können, wird ein Zugang dazu benötigt. Für das Sammeln der IP-Adressen über embedded Content sind zusätzliche Server erforderlich. Außerdem muss der Inhalt auf vielen Webseiten eingebettet werden. DNS-Server von großen Webseiten auf ORDNS zu testen ist dagegen einfacher, da diese ohne speziellen Zugang oder zusätzlicher Server abgefragt werden können. Bei dieser Methode können bei einer bestehenden ORDNS-Liste ebenfalls Hardware-Ressourcen gespart werden.

Bei einem großen Werbenetzwerk kann die Werbung gut verteilt sein, wobei viele Benutzer Werbeblocker einsetzen. Durch das Werbenetzwerk kann mit wenig zusätzlichen Hardware-Ressourcen von unterschiedlichen Beobachtungspunkten Daten gesammelt werden. Für die Abfragen von mehreren Domains müssten nach dieser Methode aber jeweils genauso viele Bilder heruntergeladen werden. Das ist aber bei der Abfrage von vielen Domains ein Problem, denn das Laden der kompletten Webseite dauert dann zu lange, sodass der Benutzer die Webseite wahrscheinlich wechselt bevor alle Bilder fertig geladen sind. Deshalb eignet sich diese Methode nur für eine Messung mit wenigen Domains.

2.2 Public Key Infrastructure

Eine Public Key Infrastructure ist eine kryptografisch gesicherte Hierarchie.

Asymmetrische Verschlüsselungsalgorithmen, wie RSA, benutzen ein Schlüsselpaar, bestehend aus einem privaten und öffentlichen Schlüssel, die miteinander in Beziehung stehen. Der private Schlüssel wird sicher beim Eigentümer aufbewahrt, der öffentliche Schlüssel wird an jeden verteilt, der mit dem Eigentümer sicher kommunizieren möchte. Es gilt für das Schlüsselpaar folgender Zusammenhang: Eine Nachricht, die mit dem öffentlichen Schlüssel verschlüsselt ist, kann nur mit dem dazugehörigen privaten Schlüssel entschlüsselt werden. Das Prinzip funktioniert auch umgekehrt. Je nach Einsatz des privaten und öffentlichen Schlüssels können unterschiedliche Schutzziele erreicht werden:

- **Vertraulichkeit**
  Zum Verschlüsseln der Nachricht benutzt der Sender den öffentlichen Schlüssel des Empfängers. Diese kann dann nur noch mit dem privaten Schlüssel des Empfängers entschlüsselt werden, also nur vom Empfänger.
- **Authentizität**
  Zum Signieren der Nachricht benutzt der Sender seinen privaten Schlüssel. Wenn der Empfänger mit dem öffentlichen Schlüssel des Senders die Nachricht entschlüsseln kann, kam die Nachricht wirklich vom Sender.

Die öffentlichen Schlüssel müssen einmalig sicher ausgetauscht werden. Um die Identität des Schlüsseleigentümers auch über ein unsicheres Medium sicherzustellen, wird der öffentliche Schlüssel von einer vertrauenswürdigen Stelle (Certificate Authority, CA) signiert. Sie stellt nach einer Überprüfung ein Zertifikat für den Schlüssel aus. Damit kann jeder die Identität überprüfen.

Eine CA kann auch eine andere CA zertifizieren (ein CA-Zertifikat ausstellen), womit wiederum Schlüssel signiert werden können. So entsteht eine Hierarchie bzw. ein Baum von CAs mit einer Wurzel-CA. Zur vollständigen Überprüfung eines Zertifikats muss jede Signatur des Pfades überprüft werden.

Alle Zertifikate haben ein Ablaufdatum und einen Verweis auf eine Sperrliste. Sperrlisten dienen dazu ausgestellte Zertifikate wieder zurückzuziehen. Ein Zertifikat ist nur gültig, wenn das Ablaufdatum noch nicht erreicht ist, es nicht auf der Sperrliste steht und alle Signaturen des Pfades gültig sind und der Wurzel-CA vertraut wird.

2.3 Routing im Internet

2.3.1 Routing

Um Daten im Internet auszutauschen, werden IP-Pakete über ein Netzwerk von Routern verschickt. Die IP-Pakete werden anhand von Quell- und Ziel-Adresse an das Ziel geleitet. Der Weg über die Router bzw. der Prozess, wie dieser Weg ausgewählt wird, wird Routing genannt.

Die Router entscheiden selbständig und dezentral mittels Routing-Tabellen, an wen die Pakete weitergeleitet werden. In den Routing-Tabellen stehen IP-Präfixe und der Next-Hop. Ein IP-

Präfix ist der Netzwerkteil einer IP-Adresse und beschreibt eine Menge von IP-Adressen, einen IP-Adressraum. Im Next-Hop stehen die IP-Adresse des nächsten Routers und das Netzwerk-Interface. Der Router sucht zu der Ziel-IP-Adresse den längsten gemeinsamen Präfix von den IP-Präfixen aus der Routing-Tabelle aus (Longest Prefix Match) und schickt das IP-Paket an den dazugehörigen Next-Hop.

Die Routing-Tabelle kann statisch vom Administrator eingetragen werden oder die Router bauen die Tabellen selbstständig dynamisch auf. Der dynamische Aufbau hat den Vorteil, dass sie flexibel auf beispielsweise geänderte Routen durch Ausfall eines Routers reagiert werden kann. Dazu tauschen Router Informationen mit benachbarten Routern über Routing-Protokolle aus.

Es wird unterschieden zwischen Interior Gateway Protocols (IGP), dem Routing innerhalb eines Autonomen Systems, und den Exterior Gateway Protocols (EGP), dem Routing zwischen Autonomen Systemen. Beispiele für IGPs sind OSPF [35] und RIP [4]. Bei EGPs ist BGP [36] Standard.

2.3.2 Autonomes System

Ein Autonomes System (AS) ist ein Teilnetzwerk des Internets, was wiederum aus Teilnetzwerken bestehen kann. Es wird von einer Organisation bzw. Unternehmen verwaltet und agiert autonom. Der Zusammenschluss Autonomer Systeme bildet das Internet.

Ein AS wird über eine AS-Nummer (ASN), eine 32-Bit-Zahl (zu Beginn nur 16-Bit)[3], identifiziert und es sind ihm mehrere IP-Präfixe zugeordnet.

Die Internet Assigned Number Authority (IANA)[4], verwaltet alle AS-Nummern und IP-Adressen und ist für deren Vergabe zuständig, wobei sie IP-Präfixe und AS-Nummern an die Regional Internet Registries (RIR) weitergibt. RIRs sind für die Vergabe in bestimmten geografischen Gebieten zuständig. Dazu zählen:

- AFRNIC (Afrikanischer Raum)
- APNIC (Asiatischer und Pazifischer Raum)
- ARIN (Nordamerikanischer Raum)
- LACNIC (Lateinamerikanischer Raum)
- RIPE (Europäischer Raum)

Diese delegieren IP-Präfixe an Local Internet Registries (LIR). Das sind meist große Provider, Unternehmen oder Universitäten, die IP-Adressen an Endkunden oder weiteren Internet Service Provider (ISP) weiter vergeben.

Der AS-Betreiber entscheidet wie das Netzwerk innerhalb organisiert ist, zum Beispiel welches Routing-Protokoll eingesetzt wird. Zwischen den Autonomen Systemen wird der Standard BGP benutzt.

[3] Um eine ASN-Knappheit zu vermeiden, wurde sie von 16-Bit auf 32-Bit vergrößert [43].
[4] https://www.iana.org/

2.3.3 Border Gateway Protocol

Das Border Gateway Protocol (BGP) [36] ist ein Routing-Protokoll der EGPs. Es tauscht Routing-Informationen zwischen Autonomen Systemen aus (EBGP). Zusätzlich kann BGP auch innerhalb eines AS benutzt werden (IBGP).

BGP ist ein Pfadvektorprotokoll und basiert auf dem Distanz-Vektor-Routing. Bei Pfadvektorprotokollen wird zusätzlich zum Distanz-Vektor der Pfad gespeichert, um Schleifen zu vermeiden. Ein Router speichert in der RIB (Routing Information Base) einen Vektor bzw. Pfad zu jedem anderen Router. In einem Vektor stehen die Kosten des Weges und über welchen direkten Router der Zielrouter erreichbar ist. Die Routing-Informationen zu Pfaden, die er kennt, sendet er an seine Nachbarn (BGP-Peers). Der Router wählt aus den Informationen, die er bekommt, und aus seinen bisherigen Informationen diejenige mit den kleinsten Kosten aus. Dabei können auch vorher festgelegte Richtlinien beachtet werden. Die ausgewählten Routen, die beim Weiterleiten verwendet werden, werden in FIB (Forwarding Information Base) gespeichert. Um die Übertragungsmenge zu verkleinern, wird bei BGP nur die Veränderung der Informationen weiter verschickt.

Zuerst bauen Router, die Routing-Informationen über BGP miteinander austauschen wollen, eine TCP-Verbindung (auf Port 179) auf. Darüber werden BGP-Nachrichten ausgetauscht. Anhand dieser Informationen und lokaler Richtlinien wird die Routing-Tabelle aufgebaut und ggf. BGP-Nachrichten weiter verschickt.

Es gibt folgende Nachrichten-Arten:

- **OPEN** ist die erste Nachricht, die beim Verbindungsaufbau geschickt wird. Darüber werden Verbindungsparameter ausgetauscht, wie zum Beispiel die BGP-Version, die ASN usw.
- **UPDATE** wird benutzt um Routing-Informationen auszutauschen. Nach diesen Informationen baut der Router seine Routing-Tabelle auf.
- **KEEPALIVE** wird in bestimmten Abständen gesendet, um zu signalisieren, dass der Router noch verfügbar ist.
- **NOTIFICATION** wird gesendet, wenn ein Fehler aufgetreten ist und die BGP-Verbindung geschlossen werden muss.

Eine Update-Nachricht enthält folgende Attribute:

- **Withdrawn Routes** ist eine Liste von IP-Präfixen, die zurückgezogen werden sollen.
- **Path Attributes** enthält eine Liste von Attributen:
  - **ORIGIN** gibt an, wo die Information herkommt. Mögliche Werte sind IGP (intern), EGP (extern) und INCOMPLETE (andere Quelle).
  - **AS-PATH** besteht aus Pfad-Segmenten und enthält alle AS-Nummern über deren Router die Nachricht verschickt wurde. Es gibt zwei Arten von Pfad-Segmenten: AS-SET, eine ungeordnete Liste von AS-Nummern, und AS-SEQUENCE, eine geordnete Liste. Eine AS-SET sollte aber nicht mehr verwendet werden [19].
  - **NEXT-HOP** ist die IP-Adresse des Routers, an den die Datenpakete, die diese IP-Präfixe betreffen, geschickt werden sollen.
  - und weitere optionale Attribute

- **Network Layer Reachability Information (NLRI)** enthält eine Liste von IP-Präfixen, die verbreitet werden soll.

Die Origin-ASN, die ASN zu dem der IP-Präfix gehört, steht in dem AS-PATH bei einer AS-SEQUENCE ganz rechts bzw. ganz hinten. Bei einer AS-SET kann man nicht bestimmen, welche ASN die Origin-ASN ist.

Bei Empfang einer Update-Nachricht werden alle IP-Präfixe aus NLRI in Adj-RIB-In hinzugefügt. Falls diese schon vorhanden sind, werden neue Informationen überschrieben. Adj-RIB-In enthält alle Routen, die von seinen Nachbarn empfangen wurden. Wenn in WITHDROWN ROUTES Routen enthalten sind, werden sie aus Adj-RIB-In gelöscht. Immer wenn sich die Adj-RIB-In ändert, wird der Entscheidungsprozess gestartet.

Entscheidungsprozess (Decision-Process):
Bei diesem Prozess werden die besten Routen bestimmt. Meistens sind das die kürzesten Routen, aber Provider können diese Entscheidungen beeinflussen, indem sie Richtlinien angeben, die in Policy Information Base (PIB) gespeichert werden. Der Entscheidungsprozess besteht aus folgenden drei Phasen:

Phase 1 bewertet die Priorität der Routen aus Adj-RIB-In. Dafür werden auch die Richtlinien aus PIB mit einbezogen.

Phase 2 wählt die besten Routen aus allen verfügbaren aus und fügt sie der Loc-RIB hinzu. Die Loc-RIB bildet die Routing-Tabelle.
Zuerst werden Routen aussortiert, bei denen im NEXT-HOP eine nicht erreichbare IP-Adresse steht oder bei denen im AS-PATH die routereigene ASN bereits enthalten ist (kreisende Nachricht). Aus allen Routen der Adj-RIB-In, die zum selben Ziel führen, wird dann die beste mit der höchsten Priorität ausgesucht. Die ausgewählten Routen werden in Loc-RIB gespeichert. Falls in Loc-RIB Routen zum selben Ziel schon vorhanden sind, müssen diese gelöscht werden.

Phase 3 wählt Routen aus Loc-RIB aus, die an Nachbarn verteilt werden und in Adj-RIB-Out gespeichert werden. Alle Routen die in Loc-RIB stehen werden in Adj-RIB-Out kopiert, wenn sie den Richtlinien aus PIB entsprechen. Danach wird der UPDATE-SEND-Prozess gestartet. Alle Routen, die bei Adj-RIB-Out hinzugekommen sind, werden per Update-Nachrichten an alle Nachbarn verteilt. Routen, die aus Adj-RIB-Out gelöscht wurden, werden per Update-Nachrichten zurückgezogen.

2.4 Probleme in BGP

2.4.1 BGP-Hijacking

Einige Schwachstellen im BGP, wie Man-in-the-Middle, können durch Authentifizierung und Integrität vermieden werden. Dazu wird für TCP eine Erweiterung für Authentifizierung TCP-MD5 [13] oder der Nachfolger TCP-AO [41] benutzt. Dabei wird bei jeder TCP-Nachricht ein Authentifikationscode mitgeschickt.

Das Routing im Internet kann trotzdem manipuliert werden, da der BGP-Inhalt nicht überprüft werden kann. Es gibt folgende zwei Angriffsarten [31]:

- **Prefix Hijacking**
  Ein AS kann fremde IP-Präfixe verteilen bzw. zurückziehen, obwohl die IP-Präfixe gar nicht zum AS gehören. Dabei können IP-Präfixe, die andere Autonome Systeme verwenden, oder auch IP-Präfixe, die noch nicht vergeben worden sind, entführt werden.

  Es gibt verschiedene Angriffe:

  – Der Angreifer verteilt ein IP-Präfix, der auf einen nicht existierenden Router verweist. Alle Anfragen laufen ins Leere. Der Service ist nicht mehr erreichbar.

  – Der Angreifer verteilt ein IP-Präfix, der auf das AS des Angreifers leitet. Dort kann er sich als Eigentümer ausgeben und Daten mitlesen oder er leitet den Datenverkehr über sein Netzwerk weiter an das ursprüngliche Ziel und kann so alle Daten mitlesen.

  Da Prefix Hijacking ein Problem im BGP ist, wird es auch BGP-Hijacking genannt.
- **AS-PATH Manipulation**
  Ein Angreifer kann auch den AS-PATH verändern, um so Routen bei gleichen IP-Präfixen im Entscheidungsprozess zu verbessern oder zu verschlechtern. Dazu wird der AS-PATH verlängert oder verkürzt. Es kann so passieren, dass Routen mit weniger Bandbreite benutzt werden und so ein „Stau“ entsteht bzw. Verbindungen nicht mehr erreichbar sind. Es können auch Verbindungen bevorzugt werden, für die der Betreiber mehr bezahlen muss.

Es gibt zwei prominente Beispiele für Prefix Hijacking:

- **YouTube**
  Pakistan wollte den Youtube-Verkehr für seine Bevölkerung sperren. Dazu wollte Pakistan Telecom alle Anfragen zu Youtube auf eine im eigenen Netz nicht existierende IP-Adresse umlenken. Die Idee war, einen spezifischeren IP-Präfix zu verteilen, um alle Routen von dem IP-Präfix 208.65.153.0/22, der zum AS36561 (Youtube) gehört, umzulenken, da ein genauerer IP-Präfix im Longest-Prefix-Match bevorzugt wird. Am 24.02.2008 begann Pakistan Telecom (AS17557) den IP-Präfix 208.65.153.0/24 zu verteilen. Der Upstream-Provider PCCW Global leitete aber dieses Update in das restliche Internet weiter, so dass nach kurzer Zeit der Youtube-Verkehr zu diesem IP-Präfix auf Pakistan umgelenkt wurde und für einen großen Teil der Internetnutzer weltweit Youtube nicht mehr erreichbar war [37].
- **China**
  Am 8. April 2010 hat China Telecom mehr als 50.000 IP-Präfixe (15% des Internetverkehrs) übernommen. Sie hat die IP-Präfixe als ASN 23724 im Internet verteilt und alle Anfragen über diese IP-Präfixe an ihr eigentliches Ziel weitergeleitet. Für 20 Minuten wurde der Großteil des Datenverkehrs über China umgelenkt [14]. Hier wurde kein spezifischerer IP-Präfix verteilt, sondern die IP-Präfixe mit falscher bzw. fremder Origin-ASN.

Oftmals entsteht BGP-Hijacking auch durch Fehlkonfiguration der Router [7].

2.4.2 Mögliche Lösungsansätze

Um das Problem zu lösen, werden Ansätze verwendet, die den Besitz eines IP-Präfixes bestätigen. Ein möglicher Lösungsansatz für das Problem BGP-Hijacking ist Secure-BGP (SBGP), eine BGP-Protokollerweiterung, die die IP-Adress-Vergabe durch eine Public Key Infrastructure (PKI) schützt. Diese Lösung erfordert jedoch einmal einen erhöhten Hardware-Aufwand für die Router, wie zusätzliche Rechenlast und Speicher für die Überprüfung der Zertifikate, sowie eine erhöhten Zeitaufwand für die einzelnen Überprüfungen [18].

Weitere Ansätze sind [17]:

- Secure Origin BGP (soBGP) [45], benutzt ebenfalls Zertifikate zur Überprüfung, allerdings mit einem Web of Trust[5];
- Pretty Secure BGP (psBGP) [32], benutzt zur Überprüfung eine PKI für ASN und Web of Trust für Präfixe;
- Internet Route Verification (IRV) [12], hierbei wird zur Überprüfung ein IRV-Server des Origin-AS gefragt, und
- ein auf DNS basierendes Verfahren [5], das ein AS-DNS-Record im DNS mit DNS Security Extensions (DNSSEC) benutzt.

Um einen einheitlichen Standard zu entwickeln, hat sich die Arbeitsgruppe SIDR (secure Inter-Domain Routing)[6] zusammengefunden und haben sich auf folgende Konzepte geeinigt:

- **Prefix Origin Validation**
  Das Paar IP-Präfix und zugehörige ASN wird kryptographisch geschützt. Dadurch kann durch Authentifizierung überprüft werden, ob das AS diesen Präfix im BGP verteilen darf.
- **Path Validation**
  Der komplette AS-PATH wird mit einer Signatur geschützt. Damit kann eine Veränderung durch Überprüfung der Signatur erkannt werden. Zusätzlich ist mit Path Validation auch Prefix Origin Validation möglich.

Die SIDR hat RPKI für Prefix Origin Validation standardisiert und entwickelt BGPsec [22] für Path Validation.

2.4.3 Prefix Origin Validation mittels RPKI

Die Idee von Resource Public Key Infrastructure (RPKI) [21] ist, die IP-Adress-Vergabe über eine PKI zu zertifizieren. Dadurch können die IP-Präfix-ASN-Paare (Internet-Ressource), die im BGP-Routing verwendet werden sollen, durch Zertifikate abgesichert werden.

Die Hierarchie von PKI wird bei RPKI benutzt und spiegelt den Weg der IP-Adressen-Vergabe wieder. Die Wurzel-CA entspricht der IANA. Jeder IP-Präfix und jede ASN, die sie an RIRs vergibt, wird mit einem Zertifikat signiert. Die RIRs stellen wiederum Zertifikate für jeden IP-Präfix, den sie weitergeben, aus. Ein Zertifikat enthält genau die IP-Präfixe und/oder ASN, die delegiert worden sind. Am Ende signiert der Eigentümer des Zertifikats ein Objekt (ROA),

[5] Netz des Vertrauens: Es wird sich gegenseitig vertraut. Die Echtheit der Schlüssel wird durch gegenseitiges Signieren bestätigt.

[6] https://datatracker.ietf.org/wg/sidr/charter/

das den IP-Präfix und die ASN enthält, die in BGP verteilt werden dürfen. Die ROAs werden zur Überprüfung von BGP-Update-Nachrichten weiterverwendet. Damit alle Zertifikate und ROAs überprüft werden können, werden sie auf öffentlichen Verzeichnissen gespeichert (Repository).

Es gibt folgende Zertifikate und signierte Objekte:

Ressource Zertifikat (RC) ist ein X.509-Zertifikat [11], [25] mit zusätzlichen Feldern für IP-Präfixe und ASN. Es kann IP-Präfixe und/oder ASN enthalten und weitere RC ausstellen.

End-Entity Zertifikat (EE) dient ausschließlich dem Erstellen bzw. Signieren von ROAs und Manifests und dessen Überprüfung. Es besteht eine 1:1 Beziehung. Der private Schlüssel wird genau einmal benutzt, um ein Objekt zu signieren. Dadurch ist es möglich ROAs bzw. Manifests wieder zurückzuziehen, indem das EE-Zertifikat auf die Sperrliste gesetzt wird. Außerdem können die privaten Schlüssel gelöscht werden und müssen nicht sicher gespeichert werden. Ein EE-Zertifikat zeigt die Zugehörigkeit der IP-Präfixe zu AS, nicht aber das Recht der Verteilung der IP-Präfixe im BGP-Routing. Dazu werden ROAs ausgestellt.

Route Origin Authorization Objekt (ROA) ist ein von einem EE-Zertifikat signiertes Objekt, das eine ASN und eine Menge von IP-Präfixen mit minimaler und optional auch maximaler Länge enthält und das Recht der Verteilung der IP-Präfixe belegt.

Manifest enthält eine Liste aller Dateien in einem Repository, wobei die Dateinamen und der dazugehörige Hash-Wert gespeichert werden. Die Liste ist mit einem EE-Zertifikat signiert.

Im Idealfall besitzt die IANA ein selbst-signiertes Wurzelzertifikat (RC) für den kompletten Adressbereich. Das wurde aber nicht umgesetzt. Momentan hat jedes RIR ein eigenes Wurzelzertifikat (RC) über den Adressbereich, der von der IANA an sie delegiert wurde. Für jeden IP-Präfix, den ein RIR an LIRs verteilt, wird ein RC erstellt, das mit dem Wurzel-RC des RIRs signiert ist. LIRs stellen für jeden verteilten IP-Präfix ein RC aus, das mit ihrem eigenen RC signiert ist. Der Weg der IP-Adressvergabe verläuft so weiter. Am Ende wird ein EE-Zertifikat ausgestellt. Mit diesem wird ein ROA signiert, das ASN und IP-Präfixe von dem AS enthalten, für die das AS autorisiert ist.

In der Abbildung 2.3 ist dazu ein Beispiel zu sehen. Der Provider hat vom RIR den IP-Präfix 1.2.0.0/16 bekommen und besitzt deshalb ein RC-Zertifikat mit einem vom RIR-RC signierten IP-Präfix. Der Provider benutzt ein EE-Zertifikat, um das ROA auszustellen. Das ROA besagt, dass das AS mit der ASN 1234 den IP-Präfix 1.2.0.0/16 verteilen darf.

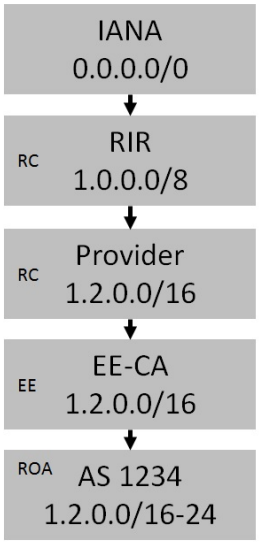


Abb. 2.3: RPKI Beispiel

Die Zertifikate und ROAs werden auf öffentlichen Servern (Repository) zur Verfügung gestellt. Jedes RIR betreibt ein Repository. Um eine Entkopplung vom RIR zu erreichen, können LIRs auch eigene Repositories eröffnen. In jedem Repository sind mindestens alle Zertifikate enthalten, die vom Betreiber signiert worden sind. Von den Repositories können alle ROAs geladen werden und alle Zertifikate, die für die Überprüfung der ROAs benötigt werden.

Da die Zertifikatsüberprüfung länger dauert bzw. mehr Hardware-Ressourcen erfordert, kann dieser Prozess auf einen vertrauenswürdigen Cache-Server im Netzwerk ausgelagert werden. Der Cache-Server lädt über „rsync“ alle ROAs herunter und überprüft diese kryptografisch. Er stellt eine Liste mit gültigen IP-Präfix-ASN-Paaren auf. Die Liste wird über das RTR-Protokoll [9] zu dem Router übertragen. Der Router speichert alle gültigen IP-Präfix-ASN-Paare in einer lokalen Datenstruktur. Beim Empfang einer Update-Nachricht kann der Router über die Datenstruktur das BGP-Update bewerten. (Siehe Abb.2.4)

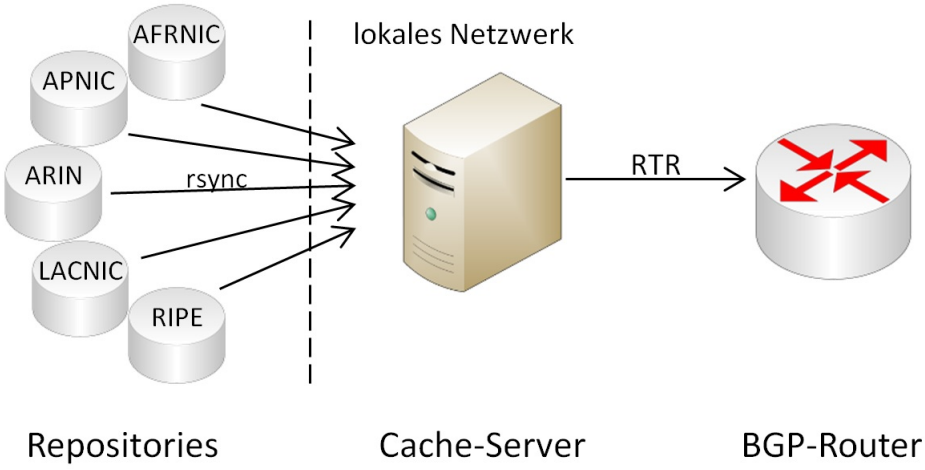


Abb. 2.4: Architektur: RPKI Cache Server

Für die Bewertung der BGP-Updates werden zu jedem IP-Präfix aus der Nachricht nach dem Longest-Prefix-Match alle ROAs gesucht, deren IP-Präfix-Länge kleiner oder gleich dem zu überprüfenden IP-Präfix ist (covering ROA).

Es gibt folgende Bewertungsstatus:

Valid Es gibt mindestens ein covering ROA, das mit ASN und IP-Präfix übereinstimmt.

Invalid Es gibt mindestens ein covering ROA und keines stimmt mit ASN und IP-Präfix überein. Zum Beispiel ist der IP-Präfix spezifischer als im max-length-Feld im ROA angegeben.

NotFound Es wurde kein covering ROA gefunden.

Mittels des Status kann der Router das BGP-Update entsprechend weiterverarbeiten.

3 Messmethode

In dieser Arbeit soll untersucht werden, inwieweit wichtige Webseiten gegen BGP-Hijacking geschützt sind, indem überprüft wird, ob die Webserver-IP-Adressen im BGP-Routing durch RPKI gesichert sind. Dazu werden für die Webserver-IP-Adressen der RPKI-Status ermittelt.

Für die Bestimmung des RPKI-Status für eine Webdomain werden jeweils IP-Präfix und ASN benötigt, die für das Routing im Internet für die Webserver dieser Webdomain benutzt werden. Mit den IP-Adressen der Webserver kann mittels Longest Prefix Match der IP-Präfix und die dazugehörige Origin-ASN aus einer Routing-Tabelle von BGP-Routern zugeordnet werden. Die IP-Adressen der Webserver von einer Webdomain können über DNS abgefragt werden. Die meisten wichtigen Webseiten benutzen CDNs [8], damit ihre Inhalte schnell und optimiert dem Endkunden zur Verfügung stehen. CDNs geben an verschiedenen Netzwerkstandorten unterschiedliche IP-Adressen für Replikationsserver zurück. Damit möglichst eine vollständige Liste aller IP-Adressen zu einer Webdomain untersucht wird, werden DNS-Abfragen an verschiedenen Netzwerkstandorten gestellt.

Die grundsätzliche Vorgehensweise ist: Für eine Webseite wird der Domain-Name über DNS an verschiedenen Standorten aufgelöst. Für die ermittelten IP-Adressen wird je ein IP-Präfix mit dazugehöriger ASN aus einer Routing-Tabelle zugeordnet. Danach wird für das IP-Präfix-ASN-Paar nach ROAs gesucht bzw. der RPKI-Status bestimmt. Falls ein gültiges ROA vorhanden ist, kann die Webseite durch RPKI geschützt werden.

Es ergeben sich so vier Schritte:

1. wichtige Webseiten bestimmen
2. DNS-Abfragen an verschiedenen Standorten
3. IP-Präfix und Origin-ASN finden
4. ROAs ermitteln

3.1 Liste wichtiger Webseiten

Für die Untersuchung muss bestimmt werden, welche Webseiten im Internet wichtige Webseiten sind. Es wird eine Liste von Domain-Namen gesucht, die alle wichtigen Webseiten enthält.

Das Unternehmen Alexa Internet Inc.[1] erstellt ein Ranking der meistbesuchten Webseiten,

[1] http://www.alexa.com/

wobei eine Liste für die ersten 1 Mio. Domains öffentlich zur Verfügung gestellt wird[2]. Alexa ermittelt den Rang anhand statistischen Daten aus einer bei ihren Nutzern installierten Browser-Toolbar. Für das Ranking werden die Besucheranzahl pro Tag und die Besucherzahlen der letzten 3 Monate betrachtet.

Diese Liste wurde auch bei anderen ähnlichen Untersuchungen verwendet [2], [3].

Alexa fasst Domains zusammen, indem alle Subdomains zur Hauptdomain gezählt werden, zum Beispiel zählt „maps.google.com“ zu „google.com“. Es gibt dabei aber auch Ausnahmen für beispielsweise einzelne Blogs. Dort werden teilweise sogar einzelne URL-Pfade zu einer Domain separat aufgelistet. In der Liste sind auch IP-Adressen enthalten, zu denen kein DNS-Name gefunden werden kann.

Für die weiteren Untersuchungen wird die komplette Liste mit 1 Mio. Einträgen verwendet. Da die Liste aber IP-Adressen und Domains mit URL-Pfad enthält, muss sie vorher folgendermaßen gefiltert und normalisiert werden:

- IP-Adressen werden heraus gefiltert und später nach den DNS-Abfragen der IP-Adress-Liste wieder hinzugefügt, da für IP-Adressen keine A- bzw. AAAA-DNS-Records mehr ermittelt werden müssen.
- Domains mit URL-Pfad müssen für die DNS-Abfragen normalisiert werden, da von einer URL nur die Domain über DNS aufgelöst werden kann. Dazu wird nur der Domain-Teil betrachtet und nur weiterverwendet, wenn dieser in der Liste noch nicht vorhanden ist und so keine Domains doppelt abgefragt werden.

3.1.1 Präfix „www“

Um die Adress-Eingabe einer URL zu verkürzen, sind Webseiten immer häufiger sowohl mit als auch ohne „www“-Präfix erreichbar (Duplicate Content), so zum Beispiel die ersten 10 Webdomains der Alexa-Liste. Die Webseiten-Betreiber verwenden dabei eine HTTP-Weiterleitung zur bevorzugten Webdomain. Es gibt aber auch Domains, die ohne „www“-Präfix nicht über DNS aufgelöst werden können (DNS-Fehler: NXDOMAIN - Domain nicht gefunden).

Um DNS-Fehler NXDOMAIN zu vermeiden bzw. um nicht jeden DNS-Server bei den DNS-Abfragen verteilt übers Internet mit einer nicht existierenden Domain abzufragen (da nicht anhand des ersten Fehlers bei einem DNS-Server entschieden werden kann, ob diese Domain tatsächlich nicht existiert oder dieser falsch konfiguriert ist), wird mit einem „Referenz“-DNS-Server vor den DNS-Abfragen entschieden, für welche Webdomains der Präfix „www“ benutzt werden muss und bei welchen der Domain-Name aus der Liste verwendet werden kann. Dazu wird mit dem „Referenz“-DNS-Server überprüft, ob die Domains aus der Liste mit oder ohne „www“ existieren. Wenn beide Domains existieren, werden beide für die DNS-Abfragen benutzt. Anderenfalls wird die erste Subdomain, die von Alexa abgefragt werden kann, getestet, damit zu diesem Rang eine Domain aufgelöst werden kann. Die Domain aus der Liste wird nicht weiterverwendet, wenn sie mit und ohne „www“-Präfix und für die erste Subdomain nicht auflösbar ist.

[2] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

3.2 DNS-Abfragen

Auf eine DNS-Abfrage gibt ein CDN an verschiedenen Netzwerkstandorten unterschiedliche IP-Adressen zurück. Damit die Untersuchung in dieser Arbeit so genau wie möglich wird, werden DNS-Server an verschiedenen geographischen und topologischen Standorten abgefragt. Das dabei auftretende Problem der verteilten Messungen von CDNs und die verschiedenen Ansätze dazu wurden bereits im Abschnitt 2.1.3 beschrieben.

Das PlanetLab-Netzwerk wird wegen der unzureichenden Verteilung der Messpunkte nicht benutzt. Eine Messkampagne kann durchaus länger als erwartet dauern, weil für eine ausreichende Verteilung der Messpunkte nicht immer genügend Benutzer verfügbar sind. Das ist zum einen für die Auswertung der Messung ungünstig, da gerade bei den zeitlich veränderlichen DNS die Antworten über einen größeren Zeitraum unterschiedlich sein können. Zum anderen lässt der zeitlich begrenzte Rahmen dieser Arbeit keine längere Messkampagne zu. Da in dieser Arbeit 1 Mio. Domains abgefragt werden sollen, kann auch der Ansatz über das Werbenetzwerk nicht benutzt werden. Deshalb werden hier ORDNS-Server verwendet. Durch eine geeignete Auswahl an Servern kann eine gute Verteilung erreicht werden. Daten stehen direkt nach den Abfragen zur Verfügung.

Die meisten CDNs wählen anhand der Quell-IP-Adresse den Replikationsserver in der Nähe des DNS-Resolvers aus. Deshalb werden verteilt über das Internet verschiedene ORDNS-Server verwendet, die durch eine rekursive DNS-Anfrage die Domain auflösen und damit den DNS-Server des CDNs anfragen. Das CDN wählt dadurch einen Replikationsserver in der Nähe des ORNDS-Servers aus [3], [15], [16], [26].

Damit eine möglichst vollständige Liste der IP-Adressen pro Domain gefunden werden kann, müssen die ausgewählten ORDNS-Server an verschiedenen Netzwerkstandorten gut verteilt sein. Dabei wird nach ASN und Land unterschieden. In dieser Arbeit wird eine bestehende Liste von ORDNS-Server benutzt, da hier kein Zugang zu Log-Dateien von großen DNS-Servern oder Webserver besteht, über die Server-IP-Adressen gesammelt werden könnten.

Das Open-Resolver-Projekt[3] sucht jede Woche nach ORDNS-Servern und stellt eine Liste von ca. 32 Mio. Servern zusammen. Aus dieser Liste werden ORDNS-Server ausgewählt, die für die DNS-Abfragen benutzt werden.

Es wird versucht, alle Domains über jeden ORDNS-Server aufzulösen. Dabei wird sowohl für IPv4 als auch nach IPv6 eine DNS-Anfrage gestellt.

3.3 Abbildung IP-Adresse zu IP-Präfix und ASN

Für die Bestimmung des RPKI-Status werden IP-Präfix und die dazugehörige ASN zu der IP-Adresse benötigt. Damit der RPKI-Status für alle Routen im Internet zu der Domain bestimmt werden kann, werden für die Zuordnung von IP-Adresse zu IP-Präfix BGP-Routing-Tabellen verwendet, da nach diesen Tabellen im Internet IP-Pakete weitergeleitet werden. Um eine große Abdeckung der Routen zu erreichen, müssen dazu mehrere BGP-Router aus unterschiedlichen

[3] Das Projekt möchte darauf hinweisen, dass Betreiber von DNS-Server darauf achten sollen, dass sie nicht auf rekursive DNS-Anfragen reagieren, um Missbrauch und DNS-Angriffe zu vermeiden. (http://openresolverproject.org/)

AS beobachtet werden.

Es gibt mehrere Router, die ihre Routing-Tabelle öffentlich bereitstellen. Diese werden Looking-Glass-Server[4] genannt. Es gibt aber auch Routing-Kollektoren, die sich mit mehreren BGP-Routern (Monitor) verbinden und deren BGP-Update-Nachrichten einsammeln und auflisten. Dabei wird angenommen, dass die Monitor-Router ihre FIB ungefiltert an den Kollektor weiter verschicken. Die Kollektoren bauen eine Routing-Tabelle mit allen IP-Präfixen auf, die ihre Monitor-Router beim Routing verwenden.

Ein Betreiber von einem AS kann seine IP-Präfixe beliebig zusammenfassen und verteilen. So ist es möglich, dass zu einem IP-Bereich zwei IP-Präfixe, ein zusammengefasster und ein spezifischer, verteilt werden. Diese können beide in unterschiedlichen Routern benutzt werden.

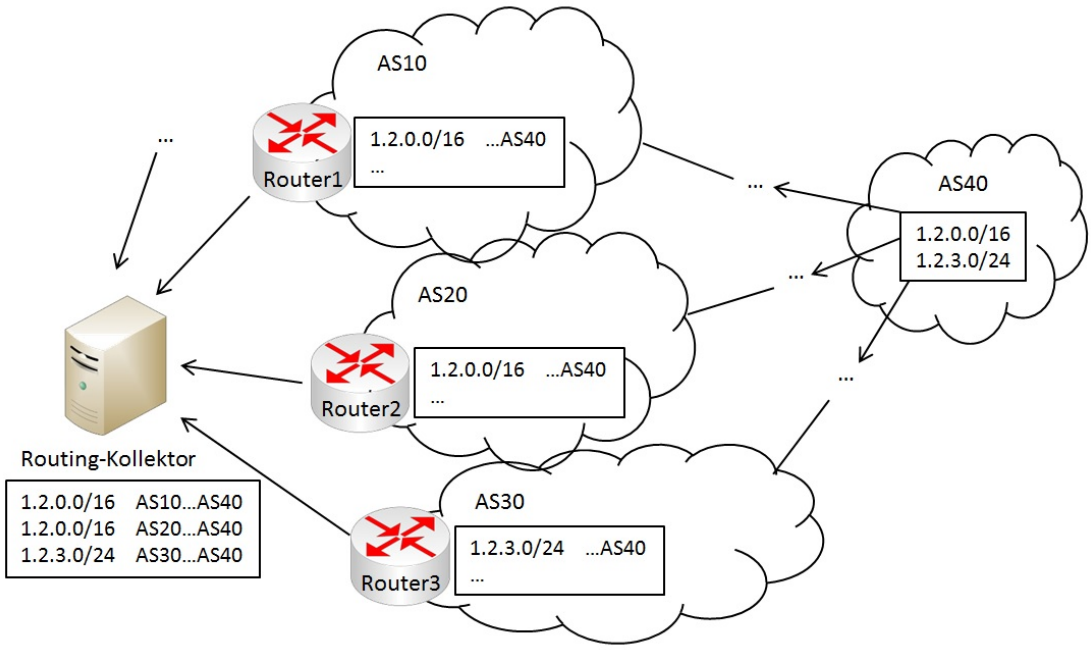


Abb. 3.1: Aufbau der Routing-Tabelle eines Routing-Kollektors

Die Abbildung 3.1 zeigt beispielhaft den Aufbau der Routing-Tabelle eines Routing-Kollektors. Der Kollektor hat je eine BGP-Verbindung zu den drei Monitorpunkten Router1, Router2, Router3 und erhält von diesen BGP-Nachrichten. In diesem Beispiel gibt es ein AS40, das einen zusammengefassten IP-Präfix „1.2.0.0/16“ und einen spezifischen IP-Präfix „1.2.3.0./24“ verteilt. Die BGP-Router erhalten die BGP-Nachrichten über mehrere Transit-Router vom AS40 und wählen den besten IP-Präfix aus und verschicken ihre Auswahl an ihre BGP-Nachbarn. Dabei erhalten Router1 und Router2 den zusammengefassten und Router3 den spezifischen IP-Präfix. Der Routing-Kollektor erhält so zweimal den zusammengefassten und einmal den spezifischen IP-Präfix.

Da Routing-Kollektoren IP-Präfixe aus mehreren Routing-Tabellen zusammenstellen, wird

[4]Liste von Looking-Glass-Servern: http://www.bgp4.as/looking-glasses

in dieser Arbeit nicht nur der längste gemeinsame IP-Präfix benutzt, sondern es werden alle IP-Präfixe zu der IP-Adresse weiter verwendet. So werden alle Monitor-Router betrachtet und auch Router, die eventuell zusammengefasste IP-Präfixe für das Routing verwenden. Bei einem Looking-Glass-Server reicht hingegen der längste gemeinsame IP-Präfix aus, da hierbei nur ein Beobachtungspunkt betrachtet wird.

In dieser Arbeit werden Routing-Kollektoren benutzt, da hierbei weniger Routing-Tabellen ausgewertet werden müssen. Dabei werden die Routing-Kollektoren von RIPE[5] verwendet. RIPE betreibt mehrere Routing-Kollektoren an Internet-Austausch-Knoten, die sich mit mehreren BGP-Peers aus verschiedenen AS verbinden. Dadurch wird eine ausreichend große Abdeckung unterschiedlicher AS erreicht. In den Routing-Tabellen (BGPdump-Dateien) der Routing-Kollektoren steht zu jedem IP-Präfix auch der komplette AS-PATH. Damit kann aus dem AS-PATH die Origin-ASN bestimmt werden. Sie steht ganz rechts im AS-PATH, wenn es eine AS-SEQUENCE ist. AS-SET werden nicht weiter betrachtet, da sie nach RFC6472 [19] nicht mehr benutzt werden sollten.

Es gibt aber in den Routing-Tabellen gleiche IP-Präfixe mit unterschiedlicher Origin-ASN. Dieses Problem wird Multiple Origin AS (MOAS) Konflikt genannt [46]. Es kann durch Fehlkonfiguration, BGP-Hijacking oder Multihoming entstehen. Bei Multihoming wird zur Ausfallsicherheit mehr als ein Provider benutzt. Da bei MOAS nicht die Origin-ASN für den Präfix eindeutig bestimmt werden kann, werden alle Origin-ASNs weiter benutzt. Bei der ROA-Suche werden für alle Origin-ASNs ROAs gesucht.

3.4 Bestimmung des RPKI-Status

Für das Suchen der ROAs nach IP-Präfix und ASN müssen alle ROAs aus allen RPKI-Repositories geladen und überprüft werden.

In einem RPKI-Repository müssen alle ROAs und Zertifikate enthalten sein, die vom Betreiber signiert worden sind. Damit alle ROAs zu einem IP-Präfix-ASN-Paar gefunden werden können, reicht es, alle ROAs aus den Repository der RIRs zu laden und kryptographisch zu überprüfen, da jedes RIR ein Repository betreibt und jedes ein selbst-signiertes Wurzelzertifikat hat und alle damit signierten ROAs enthalten muss.

Zum einen kann für das Laden das rsync-Protokoll und zur Überprüfung zum Beispiel OpenSSL verwendet werden. Zum anderen kann aber auch ein RPKI-Cache-Server benutzt werden, der die Funktionalitäten Laden und Überprüfen bereits implementiert hat und eine Liste der geprüften ROAs zur Verfügung stellt. In dieser Arbeit wird ein Cache-Server benutzt, da auch BGP-Router zum Überprüfen der BGP-Nachrichten durch RPKI einen Cache-Server verwenden.

Damit für jedes IP-Präfix-ASN-Paar ein RPKI-Status ermittelt werden kann, wird die Liste der geprüften ROAs vom Cache-Server geladen und über eine geeignete Datenstruktur werden Covering ROAs gesucht.

[5] http://www.ripe.net/data-tools/stats/ris/ris-raw-data

4 Implementierung

In diesem Abschnitt wird beschrieben, wie die vier einzelnen Teilschritte der Messmethode (Webdomains bestimmen, DNS-Abfragen, IP-Präfix und ASN finden, ROAs ermitteln) implementiert werden.

4.1 Wahl der Skript-Sprache

Alle gemessenen Daten werden in Text-Dateien, spaltenweise durch Komma getrennt (CSV) gespeichert. CSV-Dateien sind einfach aufgebaut und können von vielen verschiedenen Programmen bearbeitet werden.

Für die Teilschritte der Messung werden jeweils Skripte verwendet.

Für die Verarbeitung mit Text (Strings) werden reguläre Ausdrücke gebraucht. Deshalb wird für dieser Arbeit Perl als Skript-Sprache ausgewählt, da gerade reguläre Ausdrücke in Perl einfach benutzt werden können. Für Perl gibt es zusätzlich eine DNS-Bibliothek und es kann C-Code eingebunden werden, sodass eventuelle C-Bibliotheken benutzt werden können.

4.2 Toolchain

Die Skripte für die einzelnen Teilschritte werden nacheinander ausgeführt. Die daraus entstehende Toolchain (Abb. 4.1) besteht aus:

- **TopSites** lädt die Liste der ersten 1 Mio. Webdomains von Alexa herunter und filtert IP-Adressen und Domains, die schon vorhanden sind, heraus. Danach werden alle Domains mit dem Referenz-DNS-Server überprüft. (Siehe Abschnitt 3.1)
- **ORDNSList** testet alle DNS-Server aus einer Liste, ob sie ORDNS sind. Dadurch kann eine Liste mit ORDNS-Servern zusammengestellt werden.
- **DNS-Query** stellt an eine Liste von ORDNS-Servern DNS-Anfragen für jede Domain für IPv4 und IPv6. (Siehe Abschnitt 3.2)
- **buildBGP** lädt den aktuellen BGPdump (Routing-Tabelle) von RIPE herunter, extrahiert IP-Präfix und Origin-ASN und baut damit einen IP-Trie auf.
- **PrefixLookup** sucht zu jeder IP-Adresse alle IP-Präfixe und zugehörige ASN aus dem IP-Trie. (Siehe Abschnitt 3.3)

- **SearchROAs** sucht für jedes IP-Präfix-ASN-Paar nach ROAs und bestimmt den RPKI-Status. (Siehe Abschnitt 3.4)

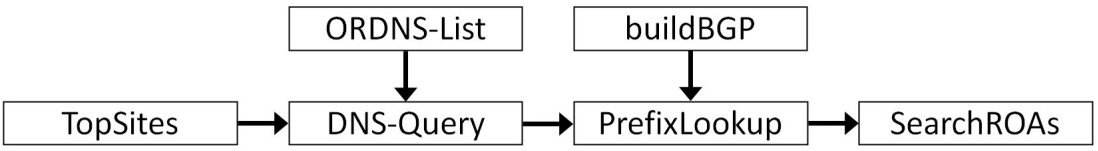


Abb. 4.1: Toolchain

4.3 Filterung der Domain-Liste von Alexa

Die Liste von Alexa enthält IP-Adressen und Domains mit URL-Pfad. Da diese so nicht weiter verwendet werden können, wird sie gefiltert und normalisiert. Dazu werden folgende reguläre Ausdrücke für IP-Adresse und Domain verwendet:

```
1 $IPv4 = '(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.)
      {3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)';
2 $IPv6 = '([0-9a-fA-F]{0,4}(:[0-9a-fA-F]{0,4})+)';
3 $IP = qr/(($IPv4)|($IPv6))/;
4 $Prefix = qr/($IP\/\d+)/;
5 $Domain = qr/([A-Za-z0-9-_]+(\.[A-Za-z0-9-_]+)*(\.[A-Za-z0-9-_]{2,}))/;
```

Quellcode 4.1: reguläre Ausdrücke

Der folgende Code-Ausschnitt 4.2 zeigt, wie die Liste der Domains von Alexa gefiltert wird.

```
1  #  -Domain mit URL-Pfad herausfiltern und nur die Domain eintragen, falls nicht
       schon vorhanden
2  #  NEU->neue Liste, IPS->nur IPs, DOPL->alle doppelten Einträge,die nach dem
       Filtern schon vorhanden sind, AUS->aussortierte Domains
3  if($domain =~ m/(^$RegEx::All::IP$)|(^$RegEx::All::IP\/.*)/){# IPs extra speichern
4    if($domain =~ m/^$RegEx::All::IP$/){
5      $ips{$domain} = 1;
6      print IPS sprintf("%s,%s,%s\n",$domain,$domain,$rank);
7      next;
8    }
9    my ($filtdomain) = $domain =~ /^($RegEx::All::IP)\/.*/;
10   unless(defined $filtdomain){
11     print "FEHLER IP: $domain\n";
12     next;
13   }
14   unless(exists $ips{$filtdomain}){
15     $ips{$filtdomain} = 1;
16     print IPS sprintf("%s,%s,%s\n",$filtdomain,$filtdomain,$rank);
17   }else{
18     print DOPL sprintf("%s,%s\n",$rank,$domain);
19   }
20 }elsif($domain =~ m/^$RegEx::All::Domain$/){# gültige Domain einfach übernehmen
21   $domains{$domain} = 1;
22   print NEU sprintf("%s,%s\n",$rank,$domain);
```

```
23  }
24  else{# ungültige Domain ->gültigen Teil herausfiltern, Test ob schon vorhanden
25    my ($filtdomain) = $domain =~ /^($RegEx::All::Domain)\/.*/;
26    unless(defined $filtdomain){# ungülitge Domain und kein gülter Teil gefunden
27      #print "FEHLER: $domain\n";
28      print AUS sprintf("%s,%s\n",$rank,$domain);
29      next;
30    }
31    unless(exists $domains{$filtdomain}){
32      $domains{$filtdomain} = 1;
33      print NEU sprintf("%s,%s,%s\n",$rank,$filtdomain,$domain);
34    }else{
35      print DOPL sprintf("%s,%s\n",$rank,$domain);
36    }
37  }
```

Quellcode 4.2: Filterung der Webdomains von Alexa

4.4 Referenz-DNS-Abfrage

Es wird erst mit einer Referenz-DNS-Abfrage überprüft, welche Webdomains aus der Liste von Alexa mit oder ohne „www“ existieren.

Als Referenz-DNS-Server wird Google-DNS[1] ausgewählt. Google-DNS beantwortet DNS-Anfragen schneller, überprüft Anfragen und Antworten und benutzt keine Filterung oder Weiterleitung. Einige lokale DNS-Server leiten zum Beispiel bei einem DNS-Fehler „NXDOMAIN-Domain nicht gefunden“ auf eine spezielle IP-Adresse um, die eine bestimmte Fehler-Seite anzeigt. Dieses Verhalten würde die Referenz-DNS-Abfrage verfälschen.

Der folgende Code-Ausschnitt beschreibt den Testvorgang. (Die Methode „queryDNS“ stellt eine DNS-Anfrage.)

```
1  # testet Domain mit und ohne www, falls beide nicht auflösbar sind, dann subdomain
       probieren
2  #  gibt Liste aller gültigen Domains zurück
3  sub testDomains{
4    my $dnsresolver = shift(@_);
5    my $rank = shift(@_);
6    my $domain = shift(@_);
7    my @domains = ();
8
9    # ohne www
10   my $dnserr = queryDNS($dnsresolver,$rank,$domain,'A');
11   unless(defined($dnserr)){
12     push(@domains,$domain);
13   }
14
15   # mit www
16   my $dnserrwww = undef;
17   unless($domain =~ /^www/){# nur wenn nicht schon www vorhanden war
18     my $wwwdomain = join('.','www',$domain);
```

[1] https://developers.google.com/speed/public-dns/

```
19      $dnserrwww = queryDNS($dnsresolver,$rank,$wwwdomain,'A');
20      unless(defined($dnserrwww)){
21        push(@domains,$wwwdomain);
22      }
23    }else{#da domain bereits bei ohne getestet wurde, hier nur der Fehler kopiert,
          damit im Fall nicht auflösbar Subdomain getestet wird
24      $dnserrwww = $dnserr;
25    }
26
27    # falls beide nicht auflösbar, dann Bubdomain testen
28    if(defined($dnserr) and defined($dnserrwww)){
29      my $dnserrsub = undef;
30      if(($dnserr eq 'NXDOMAIN' or $dnserr eq 'NOERROR') and ($dnserrwww eq '
            NXDOMAIN' or $dnserrwww eq 'NOERROR')){
31        my $subdomain = getSubDomain($domain);
32        if(defined $subdomain){
33          $dnserrsub = queryDNS($dnsresolver,$rank,$subdomain,'A');
34          unless(defined($dnserrsub)){
35            push(@domains,$subdomain);
36          }
37        }else{
38          $dnserrsub = "SUBUNDEF";
39        }
40      }else{
41        $dnserrsub = "NOTSUB";
42      }
43      if(defined($dnserrsub)){#subdomain war ok
44        printError(sprintf("%s,%s,%s,%s,%s\n",$rank,$domain,cutErrorStr($dnserr),
              cutErrorStr($dnserrwww),cutErrorStr($dnserrsub)));
45      }
46    }
47
48    return \@domains;
49  }
```

Quellcode 4.3: Referenz-Abfrage: Domain mit oder ohne www-Präfix

4.5 DNS-Bibliothek

Für DNS-Abfragen wird die DNS-Bibliothek Net::DNS von Perl verwendet.

Nach einigen Tests wurde ein Speicherfehler in der DNS-Bibliothek festgestellt. Der Speicher wächst bei jeder DNS-Abfrage stetig an. Dafür wurde aber ein Patch[2] gefunden, der das Problem löst.

Die folgende Methode wird für alle DNS-Abfragen (auch in anderen Skripten) benutzt:

```
1  # sendet eine DNS-Anfrage über den $dnsresolver
2  #  gibt eine Liste mit IPs zurück. (Liste ist leer, wenn im Antwortteil kein
       Eintrag zum gesuchten Recordtype)
3  sub sendQuery{
4    my $dnsresolver = shift(@_);
```

[2] https://puck.nether.net/ jared/raw-dns-scan/net-dns-0.72-mem-leak.patch

```
5     my $domain = shift(@_);
6     my $recordtype = shift(@_);
7
8     my @resultlist = ();
9     my $query = $dnsresolver->send($domain,$recordtype);
10    if(defined $query and defined $query->answer and scalar($query->answer)){
11      foreach my $dnsRR ($query->answer) {
12        unless ($dnsRR->type eq $recordtype){
13          next;
14        }
15        push(@resultlist,$dnsRR->address);
16      }
17    }
18    return \@resultlist;
19  }
```

Quellcode 4.4: Methode zur DNS-Anfrage

4.5.1 Parallele DNS-Abfragen

Da sehr viele DNS-Abfragen gesendet werden, ist es sinnvoll die Abfragen zu parallelisieren. Dies geschieht auf zwei Ebenen: Einerseits werden im DNS-Skript mehreren Threads benutzt, andererseits wird das Skript auf mehreren Computern ausgeführt.

Im DNS-Skript wird das Master-Worker-Modell benutzt. Es gibt ein Haupt-Thread, der die abzufragenden Domains in eine Queue schreibt, und mehrere Worker-Threads, die jeweils eine Domain aus der Queue nehmen und die DNS-Abfrage stellen. Die Ergebnis-Daten werden über ein Lock synchronisiert gespeichert. Es wird zusätzlich eine Semaphore für das Schreiben in die Queue benutzt, damit nicht am Anfang sofort alle abzufragenden Domains in der Queue und damit im Speicher stehen (speicherschonend).

```
1  # Threads initialisieren
2  my $semaphore = Thread::Semaphore->new($SemaphoreSize);
3  my $queue = Thread::Queue->new();
4  my $lock :shared;
5  my @threads = ();
6  foreach my $thrnr (1 .. $ThreadSize){
7    my $thr = threads->create(\&doWork,$thrnr,$ordnslist);
8    push (@threads, $thr);
9  }
10
11 # Bearbeitung beginnen
12 for(@tasks){
13   $semaphore->down();
14   $queue->enqueue($_);
15 }
16
17 # Worker-Threads fertig signalisieren
18 $queue->end();
19
20 for(@threads){ # auf Ende aller Threads warten
21   $_->join();
22 }
23
```

```
24 # Worker-Thread
25 sub doWork{
26   my $thrnr = shift(@_);
27   my $ordnslist = shift(@_);
28   my $dnsresolver = Net::DNS::Resolver->new;
29   while(defined(my $task = $queue->dequeue())){
30     my ($rank,$domain,$asn,$geo) = split(',',$item);
31     # DNS-Abfrage ... Ergebnisse werden über printResult() gespeichert
32     $semaphore->up();
33   }
34 }
35
36 # schreibt in Ergebnis-Datei
37 sub printResult{
38     lock($lock);
39     print RESULT $_[0];
40   RESULT->flush();
41 }
```

Quellcode 4.5: Master-Worker-Modell im DNS-Skript

Damit das Skript auf mehreren Computern gestartet werden kann, wird die Domain-Liste aufgeteilt. Es gibt dazu drei kleine Skripte: split, cont, cat

- **split** teilt die Ausgangsliste in mehrere Dateien auf und markiert sie als TODO, indem im Dateinamen TODO angefügt wird.
- **cont** sucht eine TODO-Datei, sperrt diese mit flock und markiert sie als DOING. Die Datei wird dem DNS-Skript übergeben und das Skript gestartet. Das DNS-Skript erstellt für die Ergebnisse eine neue Datei, die mit DONE markiert ist. Dieser Prozess wird solange ausgeführt bis es keine TODO-Dateien mehr gibt. Wenn es keine TODO-Dateien mehr gibt, wird im Cont-Skript überprüft, ob alle Dateien fertig bearbeitet sind. Erst danach kann das cat-Skript ausgeführt werden. Dazu wird nach der Bearbeitung einer Datei der Dateiname in die FINISHED-Datei geschrieben. Beim Test, ob alle Computer mit der Bearbeitung fertig sind, wird überprüft, ob alle DONE-Dateien aus dem Filesystem in der FINISHED-Datei stehen. Da nur ein Computer das cat-Skript ausführen kann, wird dieser am Ende über eine Datei „lastsync“ mittels flock ermittelt.
- **cat** führt alle DONE-Dateien zusammen.

Im Folgenden ist ein Auszug vom cont-Skript zu sehen.

```
1  while(defined(my $file = findFile())){#solange es keine $todo-Datei mehr gibt
2    system("perl $dnsskript $file");
3  }
4  print "Alle Dateien bearbeitet\n";
5
6  # Letzter muss Merge-Skript ausführen
7  if(allFinished()){# alle sind fertig mit bearbeiten
8    print "alle fertig!\n";
9    # Synchronisierung zwischen den Letzten über die $lastsync-Datei
10   unless(-e $lastsync){
11     open(my $fh,">>$lastsync");
12     if(flock($fh,2|4)){
13       print "Letzter\n";
```

```
14        system("perl $lastskript $dirPath");
15        unlink($finished);
16      }
17      close($fh);
18    }
19  }
20
21  # sucht zu bearbeitende Datei, sperrt sie und benennt sie in doing um. (Sperrung
        ist dann wieder aufgehoben)
22  sub findFile{
23    my @files = glob("$suffix$todo*");
24    if(scalar(@files)){
25      for(@files){
26        open(my $fh,">>$_");
27        if(flock($fh,2|4)){
28          my $splitindex = index($_,$todo);
29          my $newname = substr($_,0,$splitindex).$doing.substr($_,$splitindex+length
              ($todo));
30          rename($_,$newname);
31          close($fh);
32          return $newname;
33        }
34        close($fh);
35      }
36    }
37    return undef;
38  }
39
40  # testet, ob alle fertig sind, indem die $finished Datei mit den DONE-Dateien
        verglichen wird
41  #   ->wenn übereinstimmt, dann sind alle fertig, sonst nicht
42  sub allFinished {
43    my @doingfiles = glob("$suffix$doing*"); #todo sollte nicht mehr sein, da
        findFile() ==undef
44    my @donefiles = glob("$suffix$done*");
45    return 0 unless(scalar(@doingfiles) == scalar(@donefiles));
46
47    if(-e $finished){
48      open(my $fh,">>$finished");
49      if(flock($fh,2)){
50        open(FINISHED,"<$finished");
51        my @finishedfile = <FINISHED>;
52        close(FINISHED);
53        foreach my $done1 (@donefiles){
54          my $contains = 0;
55          foreach my $done2 (@finishedfile){
56            chomp($done2);
57            if($done1 eq $done2){
58              $contains = 1;
59              last;
60            }
61          }
62          close($fh);
63          return 0 unless ($contains);
64        }
65        close($fh);
```

```
66          return 1;
67        }
68        close($fh);
69      }
70      return 0;
71  }
```

Quellcode 4.6: cont-Skript: Dateien suchen und übergeben

4.6 IP-Trie

Für die Suche nach IP-Präfixen und ASNs für jede IP-Adresse werden die Routing-Tabellen von RIPE benutzt. Diese stellt RIPE als BGPdump-Dateien bereit. Damit aus diesen IP-Präfix und Origin-ASN gelesen werden können, wird die Bibliothek libbgpdump verwendet.

Die C-Bibliothek libbgpdump[3] wird von RIPE bereitgestellt. Sie kann BGPdump-Dateien von Zebra[4] oder Quagga[5] und MRT[6]-Dateien lesen und sie in einem einheitlichen Format ausgeben. Die Bibliothek gibt den AS-PATH als Text im folgenden Format aus: Bei einer AS-SEQUENCE sind die einzelnen ASNs mit „ “(Leerzeichen) getrennt. Die ASNs sind bei AS-SET oder AS-CONFED-SET durch „,“ getrennt und der Bereich mit „{ , }“ eingeklammert.

Die ausgelesenen IP-Präfixe werden in einer IP-Trie-Datenstruktur gespeichert, da ein Trie für die Suche nach Präfixen optimiert ist.

In den vorhandenen Perl-Bibliotheken und bei CPAN[6] gibt es keine schnell und zuverlässig funktionierende IP-Trie-Implementierung, die für IPv4 und IPv6 funktioniert und die alle IP-Präfixe zu einer IP-Adresse finden kann. Ausprobiert wurde Net::IPTrie, die zuverlässig funktioniert, aber langsam arbeitet, und Net::IP::LPM, die viel schneller ist, aber auch falsche IP-Präfixe findet und Net::Patricia, die gut und schnell funktioniert, aber nicht einfach anzupassen ist, um alle IP-Präfixe zu finden.

In dieser Arbeit wird anhand von Ansätzen aus einem Beitrag von PerlMonks[34] eine eigene Trie-Implementierung entwickelt. Sie benutzt ein BTREE von Berkeley-Datenbank[7]. Das ist eine Key-Value-Datenstruktur, bei der auch mit einem Teil von einem Key gesucht werden kann. Dieses Feature wird benutzt um ein IP-Präfix zu einer IP-Adresse zu finden. Dazu wird jeder IP-Präfix binär in einem speziellen Format gespeichert: Es wird jeweils abwechselnd ein Bit von der IP-Adresse (Netzwerkteil) und von der Subnetzmaske genommen.

Bei der Suche wird so für jedes einzelne Bit der IP-Adresse ein Key-Value-Paar gesucht und ein IP-Präfix gefunden, der mit den gesuchten Bits übereinstimmt. Das wird solange wiederholt bis es kein genaueren IP-Präfix mehr gibt. Dadurch werden auch alle IP-Präfixe, die zu der IP-Adresse gehören, gefunden.

IPv4 und IPv6 werden zusammen in einer Datenbank gespeichert, indem vor jedem IP-Präfix ein spezielles Bit-Muster für die IP-Version gesetzt wird.

[3] https://bitbucket.org/ripencc/bgpdump/wiki/Home
[4] https://www.gnu.org/software/zebra/
[5] http://www.nongnu.org/quagga/
[6] http://www.cpan.org/
[7] http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

Der folgende Code-Ausschnitt zeigt die Suche nach einem IP-Präfix (Methode „findAll“). Mit den Methoden „zipIpMask“ und „unzipIpMask“ werden IP-Präfix und Subnetzmaske in das spezielle Format konvertiert.

```
1  # sucht zu der IP alle Präfixe, die dazu passen (0.0.0.0/0 , ::0/0 wird nicht
       gefunden)
2  #  return-Format: Array mit Hash(prefix,value)
3  sub findAll {
4    my ($self, $ip) = @_;
5    my ($type,$packedip) = addrToBin($ip);
6    my @bits      = split //,$packedip;
7
8    # cursor zurücksetzen
9      $self->{DB}->seq(my $foo, my $bar, R_FIRST);
10
11     # Suche
12     my ($key, $ok, $v, @net, $maxlookup, $c);
13   $maxlookup = ($type eq IPv6) ? 128 : 32;
14   $key = $type;
15   $c = 1;
16     for(@bits) {
17         my $lk = $key .= 1 . $_;# abwechselnde Kodierung: Mask ist immer 1
18         next if $ok and $ok =~ /^$key/; # Abkürzung
19
20         my $status = $self->{DB}->seq($lk,$v,R_CURSOR);
21     unless($status == 0 && $lk =~ /^$key/){
22         return \@net;
23     }
24     # Testen ob IP im Präfix $lk liegt
25     my ($t, $prefix, $mask) = unzipIpMask($lk);
26     if ($t eq $type && inRange($prefix,$mask,$packedip)){
27       push @net, {prefix=>binToAddr($prefix, $mask),value=>$v};
28     }
29
30         $ok = $lk;
31         $c++;
32         return undef if $c > $maxlookup;
33     }
34     # einen oder kein Präfix gefunden
35   return \@net;
36 }
37 # fügt Präfix(binär) und Mask zusammen: abwechselend je ein Bit
38 sub zipIpMask {
39   my ($type, $prefix, $mask) = @_;
40
41   my @nbits  = split(//,$prefix);
42     my @mbits  = ((1) x $mask, (0) x (scalar(@nbits) - $mask));
43     my $key    = $type;
44
45     for (0..(scalar(@nbits) - 1)) {
46         my $mbit = shift @mbits;
47         my $n    = shift @nbits;
48         my $nbit = $mbit ? $n : 0;
49         $key    .= $mbit . $nbit;
50     }
51   return ($key);
```

```
52  }
53  # extrahiert wieder Präfix und Mask
54  sub unzipIpMask {
55    my ($bits) = @_;
56    my $type = substr($bits,0,8);
57    $bits = substr($bits,8);
58    my $prefix;
59    $bits =~ s/(.)(.)/$prefix .= $2; $1/ge;
60    return ($type,$prefix,$bits);
61  }
62  # überprüft, ob $addr, im Präfix($prefix/$mask) enthalten ist
63  sub inRange{
64    my ($prefix, $mask, $addr) = @_;
65    ($addr,$prefix,$mask) = map {pack "B*",$_} ($addr,$prefix,$mask);
66    my $r = $addr & $mask;
67    return $r eq $prefix;
68  }
```

Quellcode 4.7: IP-Trie-Implementierung

4.6.1 Evaluation

Die selbst entwickelte IP-Trie-Implementierung „IPTrieDB“ wird im Folgenden mit der „Net::IPTrie“-Bibliothek verglichen. Dazu wird mit beiden Implementierungen ein IP-Trie mit 5 411 846 IP-Präfixen aufgebaut und die Zeit gemessen. Dabei wird ein BGPdump-Datei von RIPE verwendet, da diese auch später für die Untersuchung benutzt wird. Die Tabelle 4.1 zeigt, dass der „IPTrieDB“ in nur 7% der Zeit von „Net::IPTrie“ den IP-Trie aufbauen kann. Danach wurde aus einer IP-Adress-Liste von DNS-Test-Daten 1 000 000 IP-Adressen ausgewählt und für je 10, 100, 1 000, 10 000, 100 000, 500 000, 1 000 000 IP-Adressen daraus im IP-Trie nach allen IP-Präfixen gesucht und dabei die Zeit gemessen. Der Test-Lauf wurde dreimal wiederholt. Die Abbildung 4.2 stellt die Mittelwerte der Laufzeiten zum Abfragen dar. Es ist zu sehen, dass beide Implementierungen asymptotisch gleich arbeiten. Allerdings ist der „IPTrieDB“ 1,3mal schneller.

	Zeit (h)
IPTrieDB	0:05:40
NET::IPTrie	1:17:34

Tab. 4.1: Laufzeit zum Aufbauen des IP-Tries

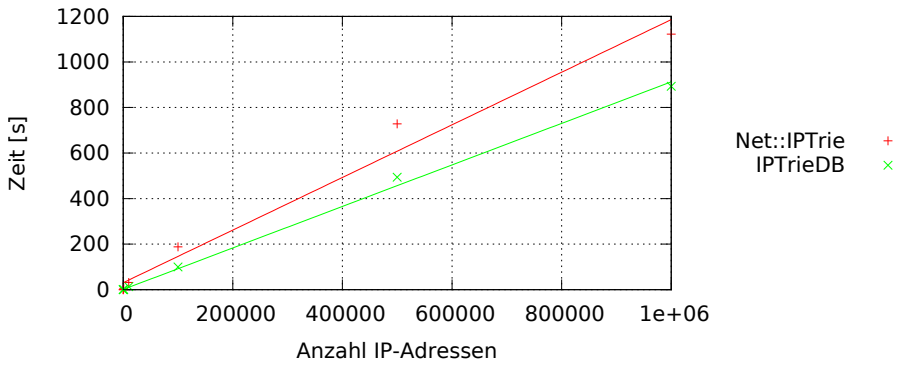


Abb. 4.2: Laufzeit zum Abfragen des IP-Tries für IPTrieDB mit Net::IPTrie

Der Vergleich zeigt, dass die selbst entwickelte IP-Trie-Implementierung schneller den IP-Trie aufbauen kann und auch etwas schneller IP-Präfixe findet.

4.7 RTR Cache-Server

Die kryptographisch geprüften ROA-Daten werden über einen Cache-Server bereitgestellt. Als Cache-Server wird der RPKI-Validator[8] von RIPE benutzt. Dieser stellt die geprüften ROAs als CSV-Datei, über JSON oder durch das RTR-Protokoll zur Verfügung.

Damit der RPKI-Status zu einem IP-Präfix-ASN-Paar ermittelt werden kann, wird die Bibliothek RTRlib[9] [27] benutzt. Diese Open-Source-C-Bibliothek implementiert das RTR-Protokoll. Sie baut aus den geprüften ROAs, die vom Cache-Server abgefragt werden, eine Datenstruktur auf, über die für ein IP-Präfix-ASN-Paar ROAs gesucht und der RPKI-Status bestimmt werden kann.

Durch die Benutzung der RTRlib wird das RTR-Protokoll als standardisierte Schnittstelle verwendet. Dadurch könnte auch jede andere Cache-Server-Implementierung benutzt werden.

Mit einer selbst entwickelten Perl-Erweiterung, die die RTRlib benutzt, kann mit einem Perl-Skript nach ROAs gesucht bzw. der RPKI-Status ermittelt werden.

4.8 Datenfluss

Bei jedem Teilschritt werden Daten aussortiert, die nicht weiter verwendet werden können oder Fehler enthalten. In Abbildung 4.3 wird eine Übersicht über den Datenfluss dargestellt.

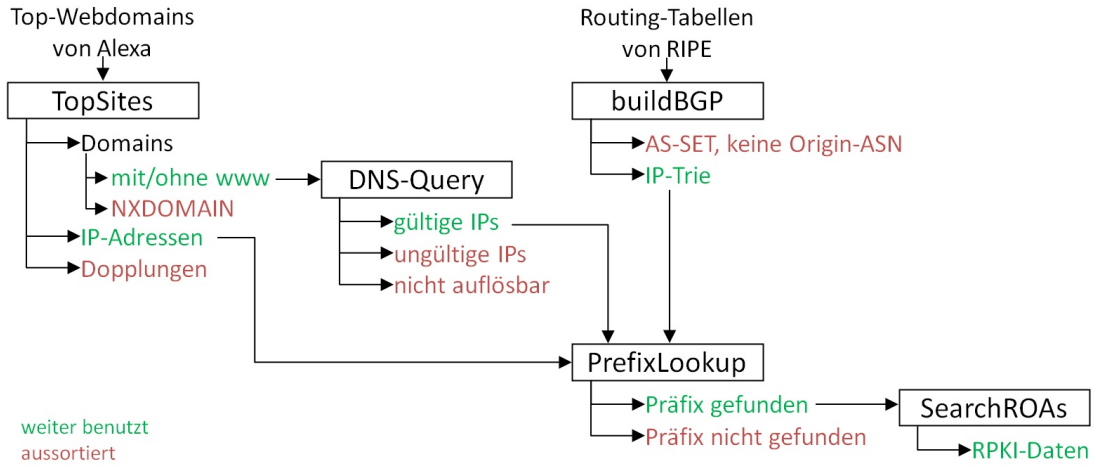


Abb. 4.3: Datenfluss

[8] http://www.ripe.net/lir-services/resource-management/certification/tools-and-resources
[9] http://rpki.realmv6.org/

Im Skript „TopSites“ werden die Top-Webdomains von Alexa heruntergeladen und gefiltert. Dabei werden IP-Adressen und Domains mit URL-Pfad, die schon enthalten sind herausgefiltert. Für die Domains der gefilterten Liste wird mit einer „Referenz“-Abfrage überprüft, ob jeweils die Domain mit bzw. ohne „www“ existiert.

Die Liste mit Domains wird im DNS-Skript verwendet. Dabei können aber einige Domains nicht aufgelöst werden und einige IP-Adressen werden herausgefiltert, da sie in reservierte IP-Bereiche fallen und damit ungültig sind.

Beim Skript „buildBGP“ werden Einträge mit einer AS-SET herausgefiltert, da dafür keine Origin-ASN bestimmt werden kann.

Der IP-Trie und die gültigen IPs und die IP-Adressen von „TopSites“ werden im Skript „Präfix-Abfragen“ benutzt. Dabei gibt es aber auch IP-Adressen, zu denen kein IP-Präfix gefunden werden konnte.

Die Liste mit gefundenen IP-Präfixen wird im Schritt „ROAs-Suchen“ weiter benutzt.

5 Zusammenstellung der ORDNS-Liste

Für die DNS-Abfragen muss vorher eine Liste von ORDNS-Servern zusammengestellt werden. Das Ziel ist eine Liste von ORDNS-Servern zu finden, die gleichmäßig verteilt sind und zuverlässig antworten. Dazu wird nach ASN und Land unterschieden. Deshalb wird zu jedem Server-Eintrag in der Liste ASN und Land bestimmt. Die Geo-Informationen werden von Maxmind[1] benutzt und die ASN wird über den selbstentwickelten IP-Trie gefunden, der mit BGP-Routing-Daten von RIPE aufgebaut wird.

5.1 Open-Resolver-Projekt

Das Open-Resolver-Projekt sucht jede Woche neu nach ORDNS-Servern und stellt eine Liste mit ca. 34 Mio. DNS-Servern bereit.

Bei einem ersten Test mit einigen Servern aus der Liste wurde festgestellt, dass die meisten nicht zuverlässig antworten. Es gibt Server, die beantworten wie erwartet rekursive Anfragen mit IP-Adresse, andere antworten mit einem DNS-Fehler (NXDOMAIN oder REFUSED), viele aber antworten gar nicht (QueryTimedOut).

Um die Zuverlässigkeit zu erhöhen, werden die wöchentlichen Listen vom 18.8.13 und 11.8.13 miteinander verglichen und nur Server-Einträge, die in beiden Listen enthalten sind, weiter benutzt. Diese Liste enthält dann nur noch ca. 9 Mio. Einträge.

Davon haben nur ca. 5 Mio. Einträge das RA-Bit gesetzt. Das RA-Bit signalisiert normalerweise dem Client, dass der Server rekursive Anfragen beantwortet. Da aber auch Server rekursiv antworten, die das RA-Bit nicht gesetzt haben, und es auch Server gibt, die das RA-Bit setzen aber nicht rekursiv antworten, wird der ORDNS-Test über das RA-Bit nicht weiterverfolgt. Vielmehr wird nun für ein ORDNS-Test eine DNS-Anfrage für „google.com“ nach IPv4 (A-Record) gestellt. Enthält die Antwort eine IP-Adresse, ist es ein ORDNS-Server. Der folgende Methode „testDnsServer“ wird für alle ORDNS-Tests benutzt.

```
1  # testet DNS-Server auf ORDNS, indem eine DNS-Anfrage für google.com nach A-Record
       gestellt wird
2  sub testDnsServer {
3    my $dnsresolver = shift(@_);
4    my @ipList = @{sendQuery($dnsresolver,'google.com','A')};
5    if(scalar(@ipList)){
6      return 1;
```

[1] www.maxmind.com

```
7     }
8     else{# leer
9       return 0;
10    }
11  }
```

Quellcode 5.1: ORDNS-Test

Da bei unzuverlässigen DNS-Servern die DNS-Anfragen länger dauern, statt sofortiger Antwort, wird die QueryTimedOut (20s) abgewartet, und dadurch nicht alle DNS-Server verwendet werden können, werden aus der Liste von gemeinsamen Server-Einträgen (9 Mio.) unterschiedliche Bereiche zufällig ausgewählt (insgesamt ca. 1,3 Mio.) und einmal auf ORDNS getestet. Davon sind 834 245 Server, die rekursive Anfragen mit IP-Adresse beantwortet haben. Die DNS-Anfragen an die ORDNS-Server werden noch weitere siebenmal wiederholt, um zuverlässige Server zu finden. Dabei ist aufgefallen, dass immer mehr DNS-Anfragen mit QueryTimedOut enden.

In der NANOG-Mailingliste[2] hat Jared beschrieben [28], dass viele CPE-Geräte[3] DNS-Anfragen an ihren lokalen DNS-Server weiterleiten und so als ORDNS fungieren. Da diese Geräte meistens eine dynamische IP-Adresse verwenden oder manchmal ausgeschaltet werden, sind so die QueryTimedOuts erklärbar.

Zur Überprüfung der Aussage von Jared werden in einer Stichprobe 100 Server, die bisher überwiegend mit QueryTimedOut endeten, zufällig ausgewählt und versucht mit dem Portscanner nmap[4] das Betriebssystem zu ermitteln, über das ein CPE-Gerät identifiziert werden kann. Dabei konnten 62 Betriebssysteme erfolgreich ermittelt werden, wovon 37 als CPE-Geräte identifiziert werden können, was die Vermutung bestätigt. Die Tabelle 5.1 zeigt die Verteilung der Betriebssysteme.

Parallel wird versucht, für die komplette Liste mit 9 Mio. Einträgen (9 728 108) den Hostname der Server über PTR aufzulösen. Es konnte bei 2 419 671 ein Hostname gefunden werden. Dabei ist auch aufgefallen, dass davon viele typische Providernamen enthalten, beispielsweise „*.t-ipconnect.de“ oder „*.dsl-w.verizon.net“. Viele Hostnames enthalten auch Zahlen von IP-Adressen mit „-“ oder „.“ getrennt, das auf Geräte mit DHCP-Adressen hinweisen könnte. Es wird versucht diese Namen über Muster zu zählen. Die Tabelle 5.2 zeigt die Verteilung. Es ist zu sehen, dass 79,6% aller Hostnames zu CPE-Geräten gehören.

Die Autoren von „On Measuring the Client-side DNS Infrastructure“ [38] haben ebenfalls festgestellt, dass 78% ihrer ORDNS-Server CPE-Geräte sind. Dazu wurden HTTP-Anfragen ausgewertet, die Spamhaus PBL[5] nach IP-Adressen der Server überprüft, und der verwendete Port überprüft. In mehreren Versuchen wurde festgestellt, dass 4% schon nach einem Tag und 80% innerhalb einer Woche nicht mehr antworteten.

[2] https://www.nanog.org/list

[3] CPE (Customer Premises Equipment) sind in diesem Kontext meist DSL-Modems, Router, NAS, W-LAN-Geräte, Firewalls... von Endkunden

[4] http://nmap.org/

[5] http://www.spamhaus.org/pbl/

Betriebssystem bzw. Geräteklasse	OS-Familie	Anzahl	CPE
Sun Integrated Light-Out Manager	embedded	12	ja
Linux	Linux	10	nein
Windows Server	Windows	8	nein
NAS device	embedded	5	ja
Cisco Switch	embedded	4	ja
FreeBSD	FreeBSD	4	nein
Linux firmware für Router, Gateway	Linux	4	ja
IPCop firewall	Linux	2	ja
IBM AIX	AIX	2	nein
DSL modem	embedded	2	ja
Cisco Router	IOS	2	ja
wireless broadband router	embedded	1	ja
Citrix NetScaler load balancer	embedded	1	ja
APC AP7920 rack PDU AOS	AOS	1	ja
Windows XP	Windows	1	ja
Avtech Room Alert environmental monitor	embedded	1	ja
Enterasys Matrix switch	embedded	1	ja
HP-UX	HP-UX	1	nein

Tab. 5.1: nmap-Test: Verteilung der Betriebssysteme

Muster	Anzahl
dsl	10 2877
cable	9 729
tele	54 741
pool	877
4 Zahlen mit „-“ oder „.“ getrennt	1 756 996
insgesamt:	1 925 220

Tab. 5.2: Verteilung der Hostname-Muster der ORDNS-Server

DNS-Server beantworten normalerweise Anfragen über Port 53. In der Liste von Open-Resolver-Projekt sind ca. 45% Einträge mit einem anderen Port enthalten. Es wird angenommen, dass DNS-Server, die auf einem anderen Port antworten, wahrscheinlich CPE-Geräte sind, die die DNS-Anfrage weiterleiten und die DNS-Antwort auf einem anderen Port zurückgeben. Deshalb werden alle Server-Einträge mit Port ungleich 53 herausgefiltert.

Parallel werden zusätzlich DNS-Server der ersten 100 000 Alexa-Domains auf ORDNS getestet. Dazu wird pro Domain eine DNS-Anfrage nach dem zuständigen DNS-Server (NS-Rekord) gestellt. Dadurch wurden 6 412 ORDNS-Server gesammelt und der Liste hinzugefügt.

Aus der gemeinsamen Liste von Open-Resolver-Projekt und DNS-Server, die über Alexa gesammelt wurden, werden nur die Server herausgefiltert, die bei allen Tests „richtig“ geantwortet haben. Es sind folgende Einträge aussortiert worden:

- Port ungleich 53 (47 Einträge)
- einmal nicht rekursiv geantwortet (55 584 Einträge)
- einmal mit QueryTimedOut (170 716 Einträge)

Es blieben 668 945 Einträge übrig.

5.1.1 Aufbau einer dynamischen Server-Liste

Bisher konnte keine feste zuverlässige Liste zusammengestellt werden. Deshalb wird ein dynamischer Ansatz benutzt. Alle DNS-Server werden nach ASN und Land gruppiert und jede Domain wird nur noch pro ASN-Geo-Paar abgefragt. Dabei wird ein Server aus der Gruppe ausgewählt. Wenn dieser nicht „richtig“ antwortet, wird der Server gewechselt und der nächste aus der Gruppe getestet.

Dazu wird eine Perl-Hash-Datenstruktur verwendet, in der alle Server-Einträge nach ASN-Geo-Paar gruppiert sind und zusätzlich der aktuelle DNS-Server aus der Gruppe gespeichert wird. Das DNS-Skript wurde wie folgt angepasst: Am Anfang wird eine Liste aller ASN-Geo-Paare eingelesen und dann alle abzufragenden Domains mit jedem ASN-Geo-Paar abgefragt. Vor und nach jeder DNS-Abfrage für ein ASN-Geo-Paar wird ein Lock gesetzt bzw. freigegeben, damit pro ASN-Geo-Paar nur ein Thread die Server abfragt bzw. testet. Vor jeder DNS-Abfrage wird der aktuelle Server aus dem ASN-Geo-Paar aus der Datenstruktur abgefragt und als DNS-Server benutzt (Methode „list->getDNSServer“). Wenn dieser zweimal mit QueryTimedOut endet oder nicht mit dem gesuchten DNS-Record-Typ antwortet, wird ein neuer DNS-Server aus dieser Gruppe ausgewählt, solange bis alle DNS-Server in der Gruppe mindestens einmal abgefragt worden sind (Methode „queryDNS“).

```
1  # Worker-Thread
2  sub doWork{
3    # ... Task aus Queue
4    # DNS-Abfrage
5    $ordnslist->lockPair($asn,$geo);
6    doDnsQuery($ordnslist,$dnsresolver,$rank,$domain,$ipv4,$ipv6,$asn,$geo);
7    $ordnslist->unlockPair($asn,$geo);
8    # ...
9  }
10
11 # queryDNS für Domain für IPv4 und IPv6 und schreibt ggf. Fehler-Datei
```

```
12  sub doDnsQuery {
13    # Parameter ...
14
15    { # aktuellen DNS-Server setzen
16      my ($dnsserver,$port) = $ordnslist->getDnsServer($asn,$geo);
17      $dnsresolver->nameservers($dnsserver);
18      $dnsresolver->port($port);
19    }
20
21    # ipv4
22    my ($dnserrv4, $triesv4) = queryDNS($ordnslist,$dnsresolver,$rank,$domain,$ipv4,
          $ipv6,$asn,$geo,'A');
23    if(defined $dnserrv4){
24      printError(sprintf("%s,%s,%s,%s,A,%s,%s\n",$rank,$domain,$asn,$geo,$triesv4,
            cutErrorStr($dnserrv4)));
25    }
26
27    # ipv6
28    my ($dnserrv6, $triesv6) = queryDNS($ordnslist,$dnsresolver,$rank,$domain,$ipv4,
          $ipv6,$asn,$geo,'AAAA');
29    if(defined $dnserrv6){
30      printError(sprintf("%s,%s,%s,%s,AAAA,%s,%s\n",$rank,$domain,$asn,$geo,$triesv6
            ,cutErrorStr($dnserrv6)));
31    }
32  }
33
34  # sendet DNS-Anfrage und schreibt bei Erfolg in Ergebnis-Datei
35  #  sonst versucht DNS-Fehler zu behandeln
36  #  gibt undef bei Erfolg, sonst letzter DNS-Fehler und #Versuche
37  sub queryDNS{
38    # Parameter ...
39
40    my $asngeoserversize = $ordnslist->getAsnGeoServerSize($asn,$geo);
41    my $dnserr = undef;
42    my $counter = 0;
43    my $serverchangecounter = 0;
44    my $originaldomain = $domain;
45    my $isPrevQuerytimedout = 0; #bool, ob vorheriger Fehler Query timed out
46    do{
47      # neue Anfrage stellen
48      $counter++;
49      my @resultlist = @{sendQuery($dnsresolver,$domain,$recordtype)};
50
51      if(scalar(@resultlist)){ #kein Fehler; Ergebnis in Ergebnis-Datei schreiben
52        for(@resultlist){
53          printResult(sprintf("%s,%s,%s,%s,%s,%s\n",$_,$domain,$rank,$asn,$geo,
                $dnsresolver->nameservers));
54        }
55        $ordnslist->setSuccess($asn,$geo);
56        return (undef,$counter);
57      }
58      # Antwort ist leer --> Fehler behandeln
59      else{
60        $dnserr = uc($dnsresolver->errorstring);
61
62        if($dnserr eq 'QUERY TIMED OUT' and !$isPrevQuerytimedout){
```

```
63          # Das erste Mal Query timed out wird gewartet, sonst Server wechseln
64          sleep(10);
65          $isPrevQuerytimedout = 1;
66        }
67        elsif($dnserr eq 'NOERROR' and (($recordtype eq 'AAAA' and not($ipv6)) or (
              $recordtype eq 'A' and not($ipv4)))){
68          # Kein DNS-Fehler und keine IP in Antwort und bei Referenz-Abfrage auch
                keine IP gefunden wurde,
69          #  dann nicht weiterversuchen
70          return (undef, $counter);
71        }
72        else{
73          # DNS-Server wechseln
74          my ($dnsserver,$port) = $ordnslist->getNewDnsServer($asn,$geo);
75          $dnsresolver->nameservers($dnsserver);
76          $dnsresolver->port($port);
77          $serverchangecounter++;
78          $isPrevQuerytimedout = 0;
79          $ordnslist->setError($asn,$geo,$dnserr);
80
81        }
82      }
83    } while($serverchangecounter < $asngeoserversize);
84    unless(defined $dnserr){
85      $dnserr = 'EMPTY';
86    }
87    return ($dnserr, $counter);
88  }
```

Quellcode 5.2: DNS-Skript mit dynamische Server-Liste

5.1.2 Ermittlung stabiler ORDNS-Server

Es wird noch eine weitere Herangehensweise ausprobiert eine zuverlässige Liste zusammenzustellen.

Es wird zum einen angenommen, dass durch eine Auswahl von ORDNS-Servern aus großen Provider-Netzwerken auch eine ausreichend gute geografische und topologische Verteilung erreicht werden kann, und zum anderen, dass CDNs mehr Infrastruktur (Replikationsserver) in große Provider-Netzwerke bereitstellen, da diese meist mehr Kunden und damit mehr Endbenutzer haben. Demnach könnten auch viele unterschiedliche IP-Adressen für Replikationsserver gefunden werden, wenn ORDNS-Server aus großen Provider-Netzwerken abgefragt werden.

Die gesamte ORDNS-Liste (9 Mio. Einträge) wird deshalb nach ASN gefiltert, die zu großen Providern gehören. Es werden zwei ASN-Listen für große Provider getestet:

- alle Tier 1 und Tier 2 Provider[6]
- die ersten 100 vom AS-Ranking[7]

[6] http://en.wikipedia.org/wiki/Tier_1_network, http://en.wikipedia.org/wiki/Tier_2_network

[7] http://as-rank.caida.org/?mode0=as-ranking&n=100&ranksort=1

Eine weitere Möglichkeit ist, aus der Server-Liste (9 Mio. Einträge) nach Hostnames zu filtern. Dazu wird nach allen Servern gesucht, die „ns“ bzw. „dns“ im Namen stehen haben, da angenommen wird, dass diese Server Nameserver für Domains sind und diese zuverlässig antworten bzw. immer verfügbar sind.

Die beiden Provider-Listen (AS-Ranking und Tier1&2) und die Hostname-Liste werden mit der bisherigen Liste verglichen. Dazu wird die Anzahl der Server und die Anzahl der ASN-Geo-Paare miteinander verglichen. Zusätzlich wird zu jeder ORDNS-Liste gezählt, zu wie vielen Server-Einträgen ein Hostname gefunden werden konnte und wie viele davon „ns“/„dns“ im Namen enthalten. Des Weiteren wird für jede ORDNS-Liste ein Testlauf mit den ersten 10 Domains durchgeführt und die dafür benötigte Zeit, die Anzahl aller gültigen IP-Adressen und die Anzahl aller DNS-Fehler verglichen.

Die bisherige Liste wird einmal ungefiltert und einmal gefiltert betrachtet. Bei der gefilterten Liste werden pro ASN-Geo-Paar min. 5 und max. 10 Server benutzt, damit bei dem dynamischen Ansatz pro Gruppe nur wenige Server ausprobiert werden müssen.

Die Tabelle 5.3 zeigt den Vergleich.

	AS-Ranking	Tier1&2	ungefiltert	gefiltert	Hostnames
ASN:	100	35			
Server:	1.709.369	486.942	668.945	36.029	37.615
ASN-Geo-Paar:	398	197	8.622	4.025	7.428
mit Hostname:	879.970	405.739	388.949	21.779	alle
(*ns*)	28.582	9.504	13.955	2.660	alle
Test für die ersten 10 Domains auf je 5 Clients:					
Zeit:	22h	18,5h	15,5h(1Client)	6h	1,5h
gültige IPs	2.792	1.304	23.911	18.123	23.194
DNS-Fehler	2.627	1.534	3.472	549	3.175

Tab. 5.3: Vergleich der ORDNS-Listen

In der Tabelle ist zu sehen, dass die Hostname-Liste gegenüber den beiden Provider-Listen mehr Server und mehr ASN-Geo-Paare enthält und annähernd so viele ASN-Geo-Paare wie die ungefilterte bisherige Liste. Da in der Hostname-Liste wahrscheinlich meistens Nameserver stehen, ist so auch die beste Zeit zu erklären. Es konnten dabei auch gleich viele gültige IP-Adressen gesammelt werden wie bei der bisherigen Liste.

Für die DNS-Abfragen wird deshalb die Hostname-Liste gewählt und der dynamischen Ansatz im Skript benutzt.

5.2 publicDNS

Die Zeit pro Domain der DNS-Abfragen, die mit der Hostname-Liste gestartet werden, verlängerte sich, da die QueryTimedOuts zunahmen und mehr DNS-Fehler entstanden. Deshalb wird

nach Alternativen gesucht.

Die Webseite public-dns[8] stellt eine Liste „Freier Nameserver“ bereit, die von jedem Benutzer verwendet werden kann. Die DNS-Server werden regelmäßig getestet. Demzufolge sollten diese DNS-Server zuverlässig DNS-Anfragen beantworten.

Von der Liste (2 860 Server) werden nur die Server benutzt, die bei einem ORDNS-Test „richtig“ antworten. Das sind 2 792 Server. Sie verteilen sich in 1 352 unterschiedlichen topologischen und geografischen Standorten (ASN-Geo-Paare).

5.3 OpenDNS

Es gibt neben Google-DNS noch einen weiteren großen bzw. weitverbreiteten DNS-Server: OpenDNS[9]. Als DNS-Server-IP-Adresse von OpenDNS wird 208.67.222.222 genutzt. Diese wird parallel zu den anderen DNS-Abfragen auch mit jeder Domain abgefragt.

[8] http://public-dns.tk/
[9] http://www.opendns.com/

6 Durchführung

In diesem Abschnitt werden die einzelnen Schritte der Durchführung beschrieben.

6.1 Referenz-Abfrage

Für die Referenz-DNS-Abfrage zur Feststellung, inwieweit Webdomains aus der Alexa-Liste mit oder ohne „www“-Präfix existieren, wird Google-DNS benutzt.

Für Google-DNS gibt es zwei IP-Adressen. Für diese Untersuchung wird 8.8.8.8 gewählt. Diese wird mit jeder Domain abgefragt und getestet. Dabei werden auch die IP-Adressen der Webdomains, die mittels DNS aufgelöst werden können, gespeichert, sodass nach diesem Durchlauf erste Ergebnisse vorhanden sind.

Am 28.09.13 wurden die aktuellen Top 1 Mio. Webdomains von Alexa geladen und nach IP-Adressen und Domains mit URL-Pfad gefiltert und normiert. Es wurden 3 152 IP-Adressen herausgefiltert und es werden 12 316 Domains, die nach dem Entfernen des URL-Pfades bereits in der Liste vorhanden waren, nicht weiter verwendet.

Die gefilterte Liste wird für die Referenz-Abfrage verwendet, die am gleichen Tag gestartet wurde. Die entstandene Liste von Domains enthält ca. 1,912 Mio. Einträge und wird weiter für die DNS-Abfragen verwendet.

6.2 DNS-Abfragen mit mehreren DNS-Resolvern

Alle DNS-Abfragen werden auf den Pool-Rechnern des Fachbereichs Informatik ausgeführt. Dabei werden nur Rechner mit Linux benutzt.

Für die Abfragen werden jeweils ca. 5-10 Pool-Rechner parallel benutzt mit jeweils 20 Threads.

Das DNS-Skript sollte mit Unterbrechungen durch beispielsweise Rechner-Neustarts umgehen können. Bei Unterbrechungen muss die gerade bearbeitete Teilliste gelöscht und nochmals ausgeführt werden. Damit der Teil, der wiederholt werden muss, relativ gering bleibt, wird die Domain-Liste in sehr kleine Teile zerlegt.

Die Domain-Liste wird zweimal aufgeteilt: Zum einen in 383 Teillisten der Größe 5000 und zum anderen, wenn eine Teilliste davon bearbeitet wird, in 1000 kleinere Teillisten mit je 5 Domains.

Die DNS-Abfragen mit der Hostname-Liste wurden am 6.10.13 gestartet. Nach einer Woche haben sich die Zeiten der Abfragen pro Domain verschlechtert. Deshalb wurde nach Alternativen gesucht und die PublicDNS-Liste am 10.10.13 parallel gestartet. Die OpenDNS-Abfrage wurde am 21.10.13 gestartet und dauerte 2,5 Tage.
Parallel dazu wurde Google-DNS noch weitere dreimal mit allen Domains abgefragt, damit später eine Veränderung der IP-Adressen pro Domain in unterschiedlichen Zeitabständen untersucht werden kann. Dazu wurde am 24.10.13, eine Woche später am 30.10. und einen Monat später am 23.11.13 die Abfrage nochmal durchgeführt.

Mit der Zeit wurden die Laufzeiten pro kleinerer Teilliste immer größer, von anfangs ein paar Stunden bis zu manchmal mehreren Tagen. Damit bei einem Abbruch der komplette Teil nicht wiederholt werden muss und so Zeit eingespart werden kann, wird das DNS-Skript so angepasst, dass der aktuelle Zustand der Abfragen jederzeit gespeichert wird und bei einem eventuellen Abbruch an der letzten Stelle weitergearbeitet werden kann.

Der Grund für die längeren Laufzeiten pro Teilliste, könnte damit zusammenhängen, dass alle DNS-Server einer Gruppe abgefragt werden, wenn bei IPv6 keine IP-Adresse in der Antwort enthalten ist. Da bei den hinteren Rängen eventuell öfter keine IPv6-Adresse existiert und deshalb öfter alle Server getestet werden müssen, könnte dies die längere Zeit verursachen. Deshalb wird das DNS-Skript angepasst, dass abgebrochen wird, wenn bei IPv6 keine IP-Adresse in der Antwort ist und auch in der Referenz-Abfrage von Google keine IPv6-Adresse existiert.

6.3 Datenübersicht nach DNS-Abfragen

Die DNS-Server Google-DNS und OpenDNS konnten je innerhalb 2,5 Tage mit allen Domains abgefragt werden. Dagegen mussten die DNS-Abfragen sowohl der Server der Hostname-Liste als auch der Server der publicDNS-Liste nach 2,5 Monate abgebrochen werden, da die Abfragen insgesamt länger als erwartet dauerten. Deshalb werden im Folgenden die beiden ORDNS-Listen getrennt von Google-DNS und OpenDNS betrachtet und ausgewertet.

Nach den DNS-Abfragen werden alle Teillisten zusammengefügt und sortiert bzw. Doppelungen entfernt. In den Daten sind zum Teil auch ungültige IP-Adressen enthalten, wie 0.0.0.0 und IP-Adressen aus den privaten Adressbereichen. Deshalb werden alle IP-Adressen herausgefiltert, die in von der IANA festgelegten reservierten IP-Bereiche für IPv4[1] und IPv6[2] fallen.

Einige IP-Adressen sind nicht „richtig“ formatiert, wie 045.006.007.086. Um Fehler zum Beispiel beim IP-Trie vorzubeugen, werden alle IP-Adressen normalisiert.

Die Tabelle 6.1 gibt eine Übersicht der gemessenen Daten.

[1] https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml
[2] https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml

	Google-DNS	OpenDNS	Hostname	PublicDNS
Zeit	2,5 Tage	2,5 Tage	2,5 Monate	2,5 Monate
abgefragt bis Rank	alle	alle	647	3611
abgefragte Domains	alle	alle	1275	5115
IPs	2 298 015	2 172 961	17 577 762	11 663 348
eindeutige IPs	2 296 859	2 172 025	1 078 655	743 463
gültige IPs	2 295 277	2 170 524	1 069 123	730 896
DNS-Fehler	-	24	1 798 130	1 143 800

Tab. 6.1: Übersicht der gesammelten IP-Adressen

6.4 Zuordnung IP-Adresse zu IP-Präfix und Bestimmung des RPKI-Status

Die IP-Adressen von Google-DNS und OpenDNS und die herausgefilterten IP-Adressen von Alexa werden zusammengeführt und alle Doppelungen entfernt. Die Liste enthält dann nur noch 2 832 628 IP-Adressen. Für die beiden anderen ORDNS-Listen wurden ebenfalls alle IP-Adressen zusammengefasst. Diese enthält dann nur noch 1 505 760 Einträge.

Für die Zuordnung von IP-Adresse zu IP-Präfix und ASN werden die Routing-Tabellen von drei Routing-Kollektoren von RIPE rr12, rrc03, rrc01, die an den drei größten Internet-Knoten DE-CIX, AMS-IX und LINX Daten einsammeln, am 19.12.13 heruntergeladen und der IP-Trie aufgebaut. Dabei konnte nicht für alle Routing-Einträge eine Origin-ASN bestimmt werden, da eine AS-SET verwendet wurde. Die Tabelle 6.2 gibt eine Übersicht der Routing-Kollektoren und zeigt jeweils die Anzahl der Routing-Einträge, der unterschiedlichen Beobachtungspunkte (nach ASN) und der Einträge mit einer AS-SET.

	rrc12 DE-CIX	rrc03 AMS-IX	rrc01 LINX
BGP-Peers	58	73	51
Einträge	8 238 913	4 244 835	5 411 995
AS-SET	270	142	149

Tab. 6.2: Übersicht der Routing-Kollektoren

Danach werden für jede IP-Adresse beider Listen alle IP-Präfixe mit zugehöriger Origin-ASN gesucht. Dabei konnten für 1052 IP-Adressen für Google-OpenDNS und 360 IP-Adressen für die ORDNS-Listen keine IP-Präfixe gefunden werden. Da danach für jede IP-Adresse ein IP-Präfix in der Ergebnis-Datei steht, werden pro Domain alle IP-Adressen mit dem gleichen IP-Präfix zusammengefasst, sodass eine Liste aller IP-Präfixe mit zugehöriger Origin-ASN pro Domain entsteht. Für Google-OpenDNS enthält sie danach 3 145 363 Einträge und für die ORDNS-Liste 460 450 Einträge.

Für beide Listen werden pro IP-Präfix-ASN-Paar der RPKI-Status bestimmt.

6.5 Datenquellen

Es werden verschiedene Datenquellen bei jedem Teilschritt benutzt. Die Abbildung 6.1 gibt dazu eine Übersicht.

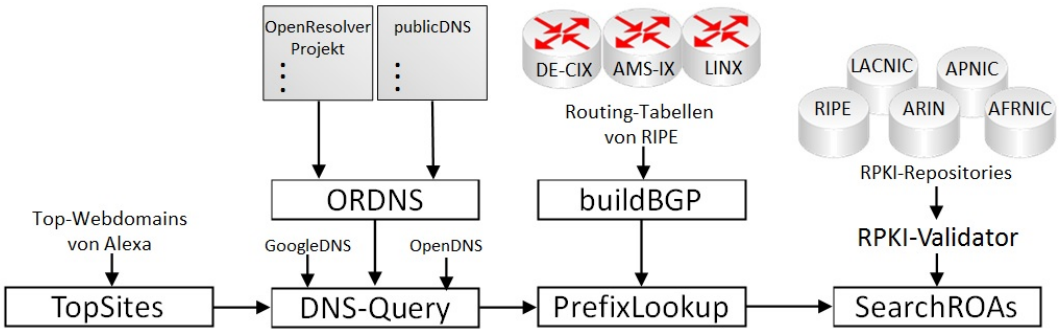


Abb. 6.1: Datenquellen für die Untersuchung

Im Skript „TopSites“ werden die Top-Webdomains von Alexa benutzt.

ORDNS-Server werden aus der Open-Resolver-Project-Liste und aus der publicDNS-Liste getestet („ORDNS“-Skript). Zusätzlich zur ORDNS-Liste wird GoogleDNS und OpenDNS im DNS-Skript benutzt.

Es werden die Routing-Tabellen von den drei Routing-Kollektoren DE-CIS, AMS-IX und LINX im „buildBGP“-Skript verwendet.

Bei der „ROA-Suche“ wird der RPKI-Validator verwendet, der alle RPKI-Daten von den 5 RIR-Repositories lädt und überprüft.

6.6 Validierung der Messmethode

Zur Überprüfung der Messmethode wird im Folgenden die Stabilität der IP-Adressen von Google-DNS untersucht. Darüber hinaus wird auch überprüft, inwieweit sich die Ergebnisse bei den DNS-Abfragen mit und ohne „www“-Präfix unterscheiden.

6.6.1 Untersuchung der Veränderung der IP-Adressen pro Domain

Google-DNS wurde insgesamt viermal mit allen Domains abgefragt, damit eine Veränderung der IP-Adressen pro Domain in unterschiedlichen Zeitabständen untersucht werden kann.

Die Abfragen werden pro Rang verglichen. Dabei wird gezählt, wie viele IP-Adressen sich unterscheiden. Am Ende wird der Gesamtdurchschnitt der relativen Häufigkeiten pro Rang ausgerechnet. Der Vergleich wird in der Tabelle 6.3 gezeigt, wobei a und b für die verglichenen Abfragen stehen.

a b	Referenz Abfr. 24.10.	Referenz Abfr. 30.10.	Referenz Abfr. 23.11.	Abfr. 24.10. Abfr. 30.10.	Abfr. 30.10. Abfr. 23.11.
Gesamtanzahl Domains	971666	970706	968752	969722	967021
Durchschnitt ungleicher IPs	0,084	0,08	0,121	0,044	0,07
fehlende Domains in a	10	9	15	993	1746
fehlende Domains in b	6608	7568	9522	1954	3694
Durchschnitt Varianz	0,068	0,0661	0,0977	0,0355	0,0591

Tab. 6.3: Vergleich der Google-Abfragen

Die Gegenüberstellung zeigt, dass sich ca. 4% der IP-Adressen innerhalb einer Woche unterscheiden, innerhalb eines Monats ca. 8% und innerhalb 2 Monate ca. 12%. Demnach verändern sich die IP-Adressen über die Zeit und je größer der zeitliche Abstand ist, desto größer ist die Veränderung.

Das bedeutet, es ist ein Problem DNS über einen größeren Zeitraum anzufragen, denn die Daten können durch die Veränderung verfälscht werden.

6.6.2 Untersuchung der Abhängigkeit vom „www“-Präfix

Für die Untersuchungen, inwieweit sich die IP-Adressen für jede Domain mit und ohne dem Präfix „www“ unterscheiden, werden die Daten von der ersten Google-DNS-Abfrage benutzt (978 274 Ränge), wobei nur die Domains verwendet werden, die mit und ohne „www“-Präfix’ auflösbar sind (937 150 Ränge).

Die IP-Adressen werden bitweise verglichen. Dabei werden IPv4 und IPv6 getrennt betrachtet. Die ersten gemeinsamen, gleichen Bits werden gezählt, einmal pro Domain und einmal insgesamt. Dafür werden die relativen Häufigkeiten gleicher Bits berechnet und pro Domain der Durchschnitt ermittelt. Auf Abbildung 6.2 ist die Verteilung gleicher Bits für IPv4 und IPv6 zu sehen. Mehr als die Hälfte der IP-Adressen sind gleich (60% bei IPv4 bzw. 70% bei IPv6). Bei IPv4 gibt es 5%-10% der IP-Adressen, bei denen weniger als 10 Bits gleich sind, bei IPv6 gibt es 5-15% der IP-Adressen, bei denen ca. 90 Bits gleich sind. In der Abbildung 6.3 werden die Domains nach Rang in 10 000er Gruppen zusammengefasst und der Durchschnitt pro Gruppe dargestellt. Sowohl bei IPv4 als auch IPv6 sind die IP-Adressen ab Rang 100 000 im Durchschnitt annähernd gleich.

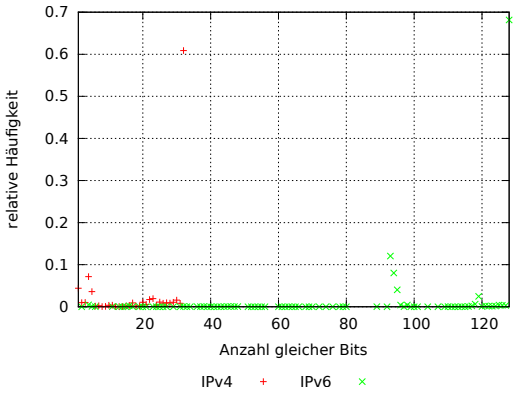


Abb. 6.2: Verteilung gleicher Bits für IPv4 und IPv6

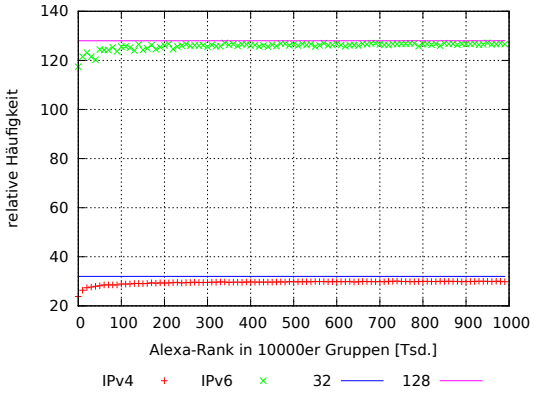


Abb. 6.3: Verteilung gleicher Bits pro Domain in 10 000er Gruppen

Die DNS-Abfragen mit oder ohne „www“-Präfix unterscheiden sich nur wenig, denn ca. 60-70% der IP-Adressen sind gleich. Wobei ein größerer Unterschied vorrangig unter den ersten 100 000 Ränge zu sehen ist und dieser für die hinteren Ränge abnimmt. Es würde demnach ausreichen ab Rang 100 000 nur noch eine von beiden Domains abzufragen.

7 Auswertung

In diesem Abschnitt werden die in der Durchführung erhaltenen Daten ausgewertet und analysiert.

7.1 RPKI-Auswertung für Google-DNS und OpenDNS

Die DNS-Server Google-DNS und OpenDNS wurden mit allen Domains aus der Referenz-Abfrage-Liste abgefragt und werden im Folgenden zusammen betrachtet.

Die Tabelle 7.1 gibt eine Übersicht über die Verteilung der RPKI-Status. Sie zeigt, dass nur bei 4% der IP-Präfixe ein gültiges ROA gefunden und für 95% keine ROAs zugeordnet werden konnte. Nur bei 0,09% der IP-Präfixe-ASN-Paare sind ROAs gefunden worden, die nicht mit IP-Präfix und ASN übereinstimmen.

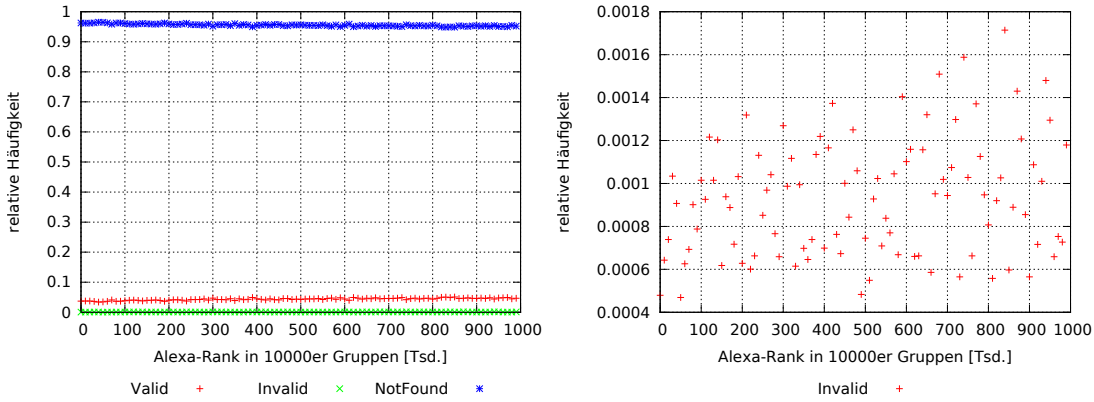
Diese Verteilung der RPKI-Status stimmt auch annähernd mit dem Ergebnis vom RPKI-Dashboard überein.

Status	Anzahl	Prozent
Valid	136 569	4,34
Invalid	2 916	0,09
NotFound	3 005 878	95,57
insgesamt	3 145 363	100

Tab. 7.1: Übersicht der RPKI-Status basierend auf GoogleDNS- und OpenDNS-Abfragen

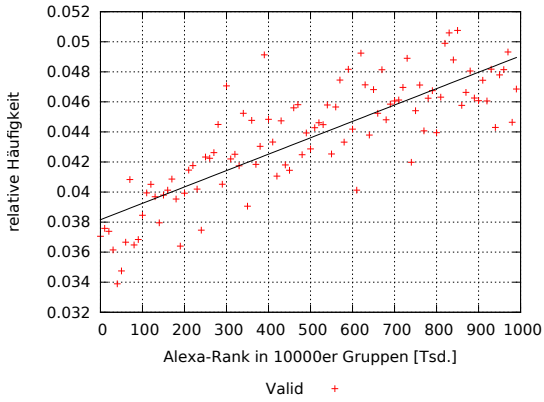
Für die Betrachtung der Verteilung der RPKI-Status pro Webseite nach Rang werden die Webdomains nach Rang in zehntausender Gruppen zusammengefasst. Dabei werden die relativen Häufigkeiten pro Gruppe für jeden Status berechnet. In Abbildung 7.1a werden alle Status zusammen dargestellt. Die Verteilung 95% „NotFound“, 4% „Valid“ und 0,09% „Invalid“ bleibt für alle Ränge annähernd gleich. Für eine genauere Untersuchung werden die Ergebnisse detaillierter skaliert in den Abbildungen 7.1b, 7.1c und 7.1d getrennt nach dem RPKI-Status dargestellt.

Bei der Verteilung nach Rang für „Invalid“ in Abb. 7.1b ist eine Streuung zwischen 0,04% und 0,12% zu sehen. Es ist keine Tendenz festzustellen.

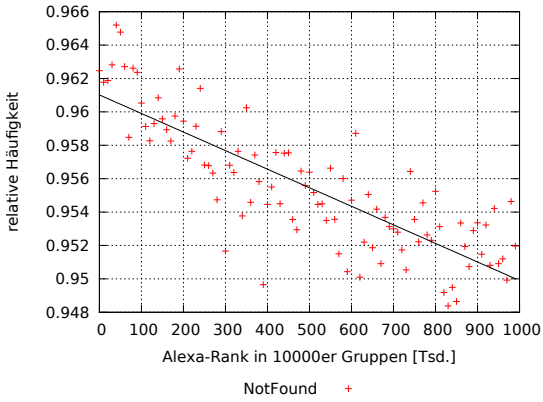


(a) Verteilung aller RPKI-Status

(b) Invalid-Verteilung



(c) Valid-Verteilung



(d) NotFound-Verteilung

Abb. 7.1: Verteilung der Prefix-Origin-Validierung nach Alexa-Rang in 10 000er Gruppen

Bei der Verteilung für „Valid“ Abb. 7.1c steigen die Werte von 3,4% für die ersten Ränge auf 4,8% für die letzten Ränge. Hierbei ist eine leicht steigende Tendenz zu beobachten, wobei sie innerhalb der ersten 300 000 Ränge stärker steigt als danach.

Für die Verteilung „NotFound“ Abb. 7.1d ist zu sehen, dass die Werte von 96,6% für die ersten Ränge auf 95% für die letzten Ränge fallen. Hier ist eine leicht fallende Tendenz erkennbar. Dabei ist innerhalb der ersten 300 000 Ränge ein stärkerer Abfall zu sehen als danach.

Die beiden Tendenzen von „Valid“ und „NotFound“ stehen in Beziehung zueinander. Da die „Invalid“-Verteilung annähernd gleichbleibend ist, ist die Tendenz von „Valid“ umgekehrt bei „NotFound“ sichtbar.

Demnach kann ein kleiner Unterschied zwischen den vorderen und hinteren Rängen festgestellt werden. Bei den ersten Rängen konnte mehr der Status „NotFound“ und bei den letzten Rängen mehr der Status „Valid“ ermittelt werden.

Webseiten der vorderen Ränge sind wichtiger eingestuft und werden häufiger besucht. Deshalb benutzen sie meist ein CDN oder stellen mehr Webserver bereit. Die Betreiber der Webseiten sind demnach abhängig von CDN-Betreibern, wenn sie ihre Webseiten durch RPKI schützen

möchten. Wenn ein CDN-Betreiber RPKI noch nicht einsetzt, dann können für alle Webseiten, die das CDN benutzen, keine gültigen ROAs gefunden werden. Allerdings könnten große Webseitenbetreiber, die ein CDN und AS gleichermaßen betreiben wie „google.com“ oder „facebook.com“, eigenständig RPKI einsetzen. Die Webseiten der hinteren Ränge verwenden meist gemietete Webserver und sind deshalb ebenfalls abhängig von ihrem Provider bzw. Anbieter, wenn sie ihre Webseite durch RPKI schützen wollen.

Zum Beleg der Aussage, dass oft CDN und Webhosting benutzt werden, werden die untersuchten IP-Präfixe nach ASN gruppiert und die Anzahl der IP-Präfixe pro AS ermittelt. Dabei zeigt die Tabelle 7.2a die ersten zehn ASN mit den meisten Vorkommen. Für die ersten zehn wurde jeweils der AS-Name bestimmt, um damit die Dienstleistung des Unternehmens zu ermitteln. In der Tabelle ist zu sehen, dass es ausschließlich CDNs oder Webhosting bzw. Internet-Hosting Anbieter sind. Damit ist die Aussage belegt.

Um die vorderen mit den hinteren Rängen vergleichen zu können, werden zudem die IP-Präfixe jeweils der ersten und der letzten zehntausend Domains nach ASN gruppiert. In der Tabelle 7.2b ist zu sehen, dass für die ersten zehntausend Domains überwiegend CDNs benutzt werden. Die Tabelle 7.2c zeigt dagegen, dass für die letzten zehntausend Domains häufiger Hosting-Anbeiter verwendet werden. Demnach verwenden wichtigere Webseiten häufiger CDNs und unwichtigere Webseiten benutzen öfter Webhosting-Angebote.

Da zu einer Domain mehrere Präfixe gehören, werden dabei einige ASN mehrfach zu einer Domain gezählt. Deshalb werden im Folgenden pro ASN alle unterschiedlichen Domains gezählt. Die ersten zehn ASNs mit den meisten Domains sind in Tabelle 7.3a dargestellt. Hierbei ist allerdings kein großer Unterschied zur ASN-Häufigkeit zu sehen.

Um ebenfalls die vorderen mit den hinteren Rängen vergleichen zu können, werden die ersten zehn ASNs mit den meisten Domains jeweils der ersten und der letzten zehntausend Domains untersucht. Die Tabelle 7.3b zeigt ebenfalls für die ersten zehntausend Domains, dass überwiegend CDNs benutzt werden. In der Tabelle 7.3c ist ebenfalls zu sehen, dass für die letzten zehntausend Domains häufiger Hosting-Anbieter verwendet werden.

Demnach besteht nur ein kleiner Unterschied in der Reihenfolge der Top-10 der ASN-Häufigkeiten gruppiert nach IP-Präfix und gruppiert nach Webdomain.

Da große Webseiten eher CDNs benutzen und kleinere Webseiten häufiger Webserver mieten und dadurch jeweils von CDN-Betreiber und Webhosting-Anbieter beim Einsatz von RPKI abhängig sind und diese RPKI noch nicht einsetzen, sind zum einen die vielen „NotFound“-Status erklärbar und zum anderen, dass der Unterschied zwischen den vorderen und hinteren Rängen nicht besonders groß ist.

Nr.	IP-Präfix		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	242835	7,72	15169	Google	u.a. CDN
2	194223	6,17	13335	CloudFlare	CDN
3	145852	4,64	26496	Go-Daddy.com	Webhoster
4	101957	3,24	46606	Unified Layer	
5	73098	2,32	36351	SoftLayer Technologies	Internet-Hosting
6	69330	2,20	24940	Hetzner Online	Webhoster
7	69081	2,20	16276	OVH Systems	Internet-Hosting
8	63805	2,03	16509	Amazon	u.a. CDN
9	57992	1,84	14618	Amazon-AES	u.a. CDN
10	39458	1,25	13768	Peer 1 Hosting	Internet-Hosting

(a) Top 10 aller IP-Präfixe der Webdomains

Nr.	IP-Präfix		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	699	6,99	20940	Akamai	CDN
2	667	6,67	15169	Google	u.a. CDN
3	646	6,46	31377	Akamai	CDN
4	641	6,41	13335	CloudFlare	CDN
5	350	3,50	16509	Amazon	u.a. CDN
6	307	3,07	14618	Amazon-AES	u.a. CDN
7	216	2,16	36351	SoftLayer Technologies	Internet-Hosting
8	201	2,01	2914	NTT America	
9	190	1,90	3320	Deutsche Telekom	
10	171	1,71	4134	ChinaNet	

(b) Top 10 aller IP-Präfixe der ersten zehntausend Webdomains

Nr.	IP-Präfix		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	859	8,59	15169	Google	u.a. CDN
2	524	5,24	26496	Go-Daddy.com	Webhoster
3	478	4,78	13335	CloudFlare	CDN
4	327	3,27	46606	Unified Layer	
5	239	2,39	36351	SoftLayer Technologies	Internet-Hosting
6	238	2,38	16276	OVH Systems	Internet-Hosting
7	234	2,34	24940	Hetzner Online	Webhoster
8	197	1,97	13768	Peer 1 Hosting	Internet-Hosting
9	188	1,88	16509	Amazon	u.a. CDN
10	186	1,86	14618	Amazon-AES	u.a. CDN

(c) Top 10 aller IP-Präfixe der letzten zehntausend Webdomains

Tab. 7.2: IP-Präfixe der Webdomains gruppiert nach ASN

Nr.	Webdomain		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	50505	5,16	46606	Unified Layer	
2	36194	3,70	26496	Go-Daddy.com	Webhoster
3	35945	3,67	15169	Google	u.a. CDN
4	32884	3,36	24940	Hetzner Online	Webhoster
5	32841	3,36	16276	OVH Systems	Internet-Hosting
6	31811	3,25	36351	SoftLayer Technologies	Internet-Hosting
7	25960	2,65	13335	CloudFlare	CDN
8	24556	2,51	16509	Amazon	u.a. CDN
9	18015	1,84	14618	Amazon-AES	u.a. CDN
10	16105	1,65	20013	CyrusOne	Internet-Hosting

(a) Top 10 aller Webdomains

Nr.	Webdomain		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	650	6,50	20940	Akamai	CDN
2	649	6,49	31377	Akamai	CDN
3	400	4,00	3320	Deutsche Telekom	
4	318	3,18	16509	Amazon	u.a. CDN
5	316	3,16	2914	NTT America	
6	287	2,87	3257	Tinet SpA	
7	239	2,39	14618	Amazon-AES	u.a. CDN
8	231	2,31	36351	SoftLayer Technologies	Internet-Hosting
9	214	2,14	4134	ChinaNet	
10	188	1,88	13335	CloudFlare	CDN

(b) Top 10 der ersten zehntausend Webdomains

Nr.	Webdomain		ASN	Name	Dienstleistung
	Anzahl	Prozent			
1	451	4,51	46606	Unified Layer	
2	373	3,73	15169	Google	u.a. CDN
3	361	3,61	26496	Go-Daddy.com	Webhoster
4	299	2,99	24940	Hetzner Online	Webhoster
5	262	2,62	16276	OVH Systems	Internet-Hosting
6	252	2,52	36351	SoftLayer Technologies	Internet-Hosting
7	164	1,64	13768	Peer 1 Hosting	Internet-Hosting
8	159	1,59	16509	Amazon	u.a. CDN
9	143	1,43	20013	CyrusOne	Internet-Hosting
10	142	1,42	8560	1&1	

(c) Top 10 der letzten zehntausend Webdomains

Tab. 7.3: Webdomains gruppiert nach ASN

7.1.1 Ursachen für Status Invalid

Der Status „Invalid“ kann entstehen, wenn zum einen die ASN mit dem Covering-ROA nicht übereinstimmt oder zum anderen ASN und IP-Präfix übereinstimmen, aber die maximale Präfixlänge überschritten wird. Im Folgenden wird das Auftreten dieser beiden Ursachen gegenübergestellt.

Ursache	Anzahl	Prozent
spezifischer IP-Präfix	1274	43,69
falsche ASN	1642	56,31
insgesamt	2916	100

Tab. 7.4: Ursachen für Invalid

In der Tabelle 7.4 ist zu sehen, dass 40% der Status „Invalid“ durch die Überschreitung der maximalen Präfixlänge und 60% durch falsche ASN verursacht wird.

Demnach stehen in den Routing-Tabellen falsche IP-Präfix-ASN-Paare und damit falsche Routing-Einträge, die durch entführte IP-Präfixe oder ASNs oder durch Fehlkonfiguration entstanden sein können. Die Überschreitungen der Präfixlängen können aber auch dadurch entstanden sein, dass ein Provider seinen IP-Präfix mit einem ROA geschützt hat, aber Teil-IP-Bereiche daraus an andere Subprovider weitergegeben hat und diese noch kein RPKI einsetzen. Dadurch sind alle BGP-Update-Nachrichten mit IP-Präfixen der Subprovider ungültig, da diese ein spezifischeren IP-Präfix enthalten, als in dem Provider-ROA angegeben ist. Solche ROA-Fehlkonfigurationen wurden bereits in der Praxis beobachtet [44].

7.2 RPKI-Auswertung für beide ORDNS-Listen

Die DNS-Server aus der Hostname-Liste und der PublicDNS-Liste konnten aus Zeitgründen (siehe Abschnitt 6.3) nicht mit allen Domains abgefragt werden. Trotzdem werden die Daten im Folgenden für die abgefragten Domains ausgewertet.

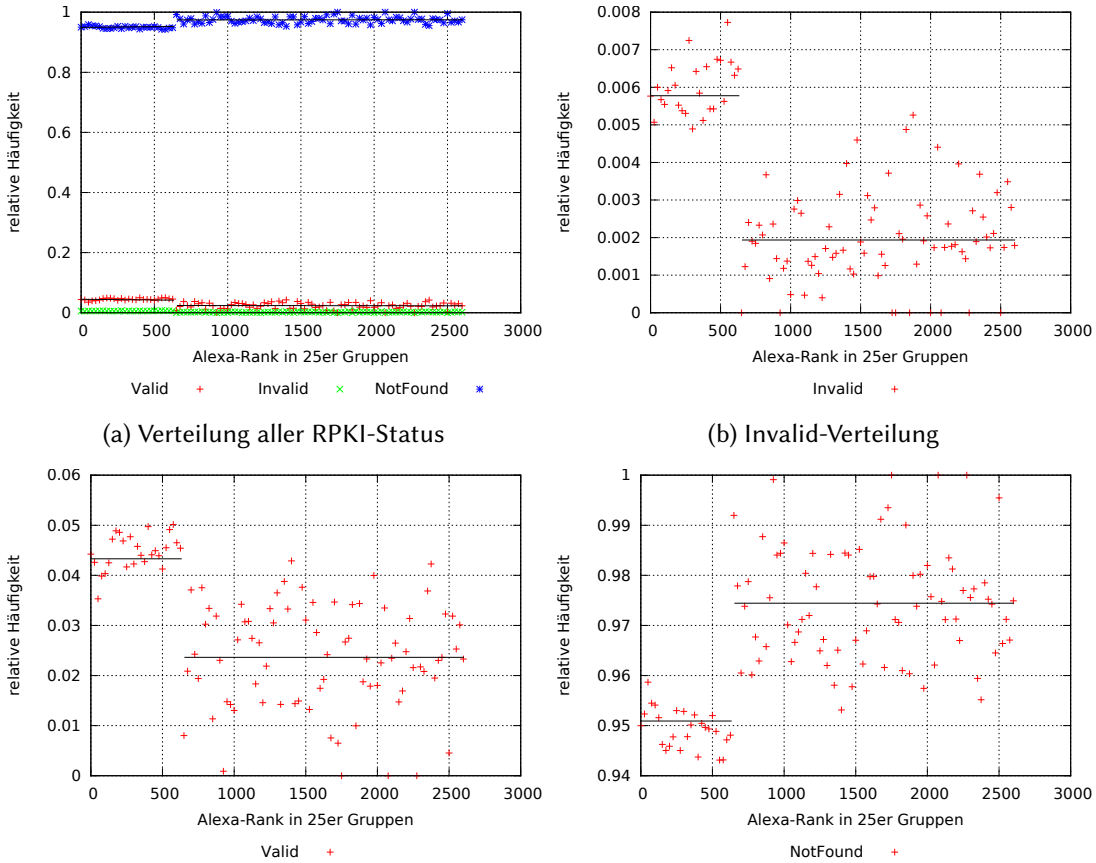
Eine Übersicht der Verteilung der RPKI-Status ist in Tabelle 7.5 zu sehen. Es konnte nur bei 3,8% der IP-Präfixe ein gültiges ROA gefunden werden und 0,5% der IP-Präfix-ASN-Paare stimmen nicht mit den Covering-ROA überein. Für die meisten (96%) konnte kein ROA gefunden werden.

Status	Anzahl	Prozent
Valid	17 472	3,79
Invalid	2 096	0,46
NotFound	440 882	95,75
ingesamt	460 450	100

Tab. 7.5: Übersicht der RPKI-Status basierend auf ORDNS-Abfragen

Diese Verteilung von der Hostname- und PublicDNS-Liste stimmt mit der Verteilung von Google-DNS und OpenDNS überein.

Es sind nur Daten für die ersten 2650 Ränge verfügbar. Deshalb werden sie für die Betrachtung nach Rang in nur 25er Gruppen zusammengefasst. Es werden ebenfalls die relativen Häufigkeiten für jeden Status pro Gruppe ermittelt. Die Abbildung 7.2a zeigt eine Übersicht aller RPKI-Status nach Rang. Hierbei ist zu sehen, dass für die ersten 650 Ränge gleichbleibend 95% den Status „NotFound“ und 4,5% „Valid“ ermittelt wurde. Ab Rang 650 beginnt eine kleine Schwankung für „NotFound“ zwischen 95% und 100% und für „Valid“ zwischen 4% und 0%. Für den Status „Invalid“ ist keine besondere Veränderung zu erkennen.



(a) Verteilung aller RPKI-Status

(b) Invalid-Verteilung

(c) Valid-Verteilung

(d) NotFound-Verteilung

Abb. 7.2: Verteilung der Prefix-Origin-Validierung nach Alexa-Rang in 25er Gruppen

Damit diese Veränderung genauer betrachtet werden kann, wird die Verteilung für jeden Status getrennt dargestellt. In Abbildung 7.2b für den Status „Invalid“ ist ebenfalls bei Rang 650 ein Schnitt zu sehen. Für die Ränge vor 650 sind ca. 0,6% „Invalid“. Für alle Ränge nach 650 ist eine Streuung zwischen den Werten 0,1% und 0,4 zu erkennen. Die Verteilung für „Valid“ (Abb. 7.2c) zeigt eine ähnliche Charakteristik. Für die ersten 650 schwankt die relative Häufigkeit um den Wert 4,5%. Für die weiteren Ränge ist eine starke Streuung zwischen den Werten 1% und 3,5% zu sehen. Die Abbildung 7.2d zeigt die Verteilung für den Status „NotFound“. Hierbei

gibt es ebenfalls einen Unterschied zwischen den Rängen vor und nach 650. Für die Ränge vor 650 konnten für 95% keine ROAs gefunden werden. Für die Ränge nach 650 ist eine Streuung zwischen den Werten 96% und 100% zu sehen.

Demnach ist für die ersten 650 Ränge eine andere Verteilung als für die Ränge danach festzustellen. Die Tabelle 7.6 zeigt die Verteilung vor und nach Rang 650. Für die ersten 650 Ränge sind demzufolge mehr gültige ROAs als für die Folgenden gefunden worden. Allerdings wurde dabei auch für mehr IP-Präfix-ASN-Paare der Status „Invalid“ bestimmt.

Status	vor Rang 650	nach Rang 650
Valid	4,5%	1-3,5%
Invalid	0,6%	0,1-0,4%
NotFound	95%	96-100%

Tab. 7.6: RPKI-Status für vor und nach Rang 650

Da die Hostname-Liste nur bis Rang 647 abgefragt wurde und damit für alle Ränge vor 647 an mehr Beobachtungspunkten gemessen wurde, kommt es zu einem Schnitt bei Rang 650. Demnach würden sich die Werte nach 650 bei mehreren Messpunkten einem Mittelwert annähern.

7.2.1 Auswirkungen eingeschränkter DNS-Abfragen

Da die DNS-Abfragen mit der Hostname-Liste und der PublicDNS-Liste länger als erwartet andauerten und sie deshalb nicht mit allen Domains abgefragt werden konnten, wird im Folgenden untersucht, inwieweit sich die Ergebnisse der beiden Listen zu den Ergebnissen von Google-DNS und OpenDNS unterscheiden.

7.2.1.1 Vergleich RPKI-Status

Es wird im Folgenden genauer der RPKI-Status der GoogleDNS-OpenDNS-Auswertung und der Auswertung der ORDNS-Server-Listen miteinander verglichen. Dazu werden von der GoogleDNS-OpenDNS-Liste nur die ersten 2650 Ränge betrachtet. Die Abbildungen 7.3 und 7.4 stellen die beiden Verteilung der RPKI-Status der ersten 2650 Ränge gegenüber. Für die Hostname-PublicDNS-Liste ist für die ersten 650 Ränge eine gleichbleibende Verteilung zu sehen und danach pro Status eine kleine Streuung. Bei der GoogleDNS-OpenDNS-Liste ist dagegen kein Unterschied zwischen den ersten 650 Rängen zu den darauffolgenden zu erkennen. Die Streuung pro Status ist dagegen stärker.

Da bei der Hostname-PublicDNS-Liste die Streuung pro Status kleiner ist, würde sich auch für die GoogleDNS-OpenDNS-Liste die Streuung einem Mittelwert angleichen, wenn mehr IP-Adressen gesammelt werden bzw. wenn ebenfalls von mehreren unterschiedlichen Beobachtungspunkte gemessen wird.

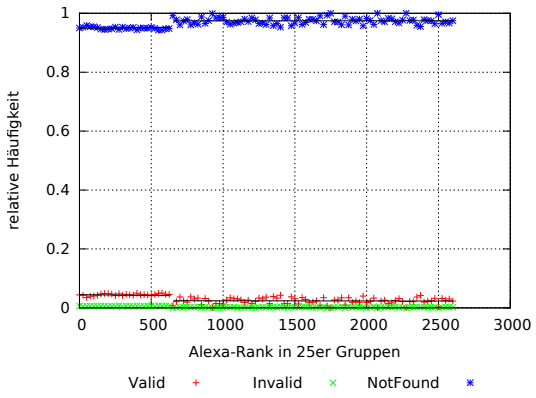


Abb. 7.3: RPKI-Status von Hostname-PublicDNS nach Rang in 25er Gruppen

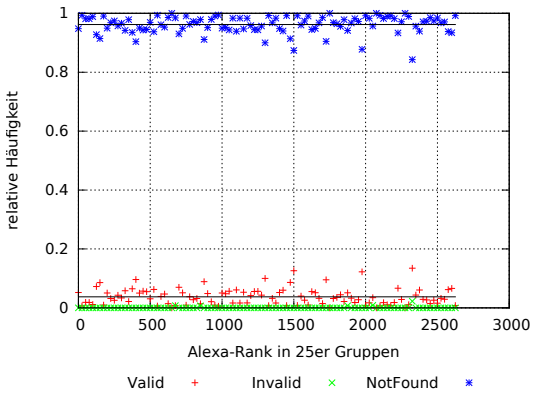


Abb. 7.4: RPKI-Status von Google-OpenDNS nach Rang in 25er Gruppen

7.2.1.2 Vergleich der IP-Adressen Google-DNS mit OpenDNS

Es wurden an zwei unterschiedlichen Netzwerkstandorten DNS-Anfragen für alle Domains gestellt, Google-DNS und OpenDNS. Die beiden Abfragen werden miteinander verglichen, damit der Unterschied nach Rang ermittelt werden kann.

Bei beiden DNS-Abfragen wird gezählt, wie viele IP-Adressen sich unterscheiden, und die relativen Häufigkeiten pro Rang berechnet. Die Abbildung 7.5 stellt die durchschnittlichen Unterschiede nach Rang in zehntausender Gruppen zusammengefasst dar. Es ist zu sehen, dass innerhalb der ersten 50 000 Ränge der durchschnittliche Unterschied vom Wert 25% stark abnimmt und danach zwischen den Werten 5% und 10% etwa gleich bleibt.

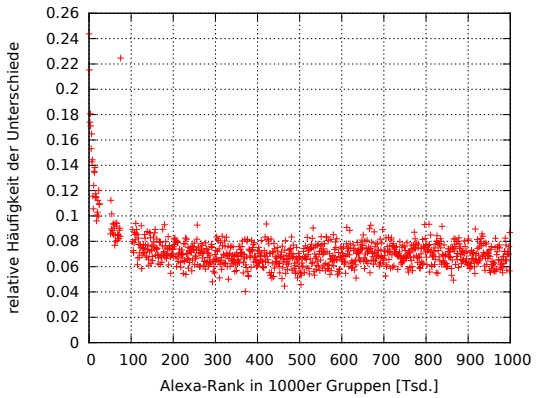


Abb. 7.5: Vergleich Google-DNS (Referenz) mit OpenDNS

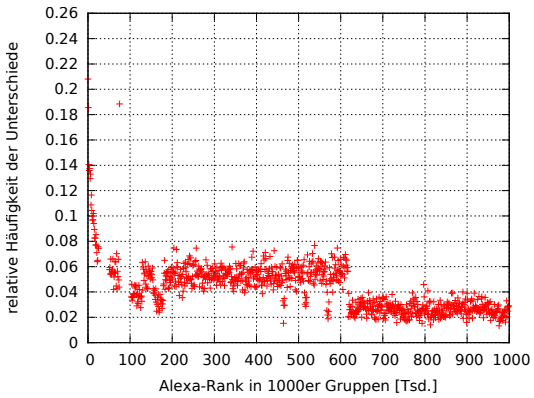


Abb. 7.6: Vergleich Google-DNS (24.10.13) mit OpenDNS

Nach der Untersuchung der zeitlichen Veränderung der IP-Adressen bei Google-DNS wird angenommen, dass der Unterschied der IP-Adressen nach Rang 50 000 auch durch den zeitlichen Unterschied entsteht. Deshalb wird OpenDNS mit der zweiten Google-Abfrage vom 24.10.13, die direkt nach OpenDNS ausgeführt wurde, verglichen. Auf Abbildung 7.6 ist zu sehen, dass der durchschnittliche Unterschied von 22% innerhalb der ersten 50 000 auf 8% sinkt und danach

gleich bleibt. Ab Rang 600 000 sinkt der Unterschied nochmals auf 2-4%.

Der Vergleich zeigt, dass sich die IP-Adressen innerhalb der ersten 100 000 Ränge unterscheiden. Ab Rang 600 000 ist der Unterschied zwischen den IP-Adressen nur noch gering. Demnach ist für die ersten 100 000 Ränge eine gute Verteilung der Messpunkte entscheidend.

8 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, den Schutz wichtiger Webseiten gegen BGP-Hijacking durch Einsatz von RPKI zu untersuchen. Dazu wurden mittels ORDNS-Servern an verschiedenen Netzwerkstandorten DNS-Abfragen für alle wichtigen Webseiten von Alexa gestellt, den ermittelten IP-Adressen wurden IP-Präfix und ASN aus Routing-Tabellen von RIPE zugeordnet und dafür je der RPKI-Status bestimmt. Die Ergebnisse zeigen, dass 95% der Webseiten ohne RPKI-Schutz betrieben werden und damit die Umsetzung ähnlich weit ist, wie für die gesamten IP-Präfixe. Eine mögliche Ursache für den fehlenden RPKI-Schutz kann die Abhängigkeit vieler Webseiten von CDN-Betreibern oder Webhosting-Anbietern sein.

Weiterhin wurde der Unterschied zwischen wichtigen und weniger wichtigen Webseiten analysiert. Dabei konnte nur ein minimaler Unterschied zwischen den Rängen der meist besuchten Webseiten festgestellt werden: Für die in der Rangliste höher platzierten Webseiten konnte minimal weniger der RPKI-Status „Valid“ ermittelt werden. Ein möglicher Grund dafür ist, dass die meisten wichtigen Webseiten der vorderen Ränge CDNs benutzen und Webseiten der hinteren Ränge öfter Webserver mieten und damit von CDN-Betreiber und Webhosting-Anbieter beim Einsatz von RPKI abhängig sind.

Um zukünftig Probleme wie den Youtube-Unfall von 2008 zu vermeiden, sollte RPKI weiter umgesetzt werden. Besonders Betreiber wichtiger Webseiten sollten daran interessiert sein, damit ihre Inhalte vor Angriffen geschützt werden können. Problematisch dabei ist, dass einige Webseiten-Betreiber von Providern, CDN-Betreiber oder Webhosting-Anbietern abhängig sind. Die IP-Präfix-Eigentümer müssen ihre Präfixe durch RPKI schützen, indem sie ROAs dafür ausstellen. Allerdings dürfen Provider, die Teil-IP-Bereiche aus ihrem IP-Präfix an Subprovidern delegiert haben, nicht ein ROA über den kompletten IP-Bereich erstellen, wenn Subprovider noch kein RPKI einsetzen. Sie sollten dann nur für nicht delegierte Teil-Bereiche ROAs erstellen, um ungültige BGP-Routen zu verhindern. Die RIRs bieten Unterstützung bei der Umsetzung, auch um Fehler bei der Erstellung von ROAs zu vermeiden. Um gegen BGP-Hijacking geschützt zu sein, reicht es nicht ROAs zu erstellen, BGP-Router müssen auch die BGP-Update-Nachrichten bewerten und die Route entsprechend weiterverarbeiten. Die IP-Präfix-Eigentümer müssen darauf aufmerksam gemacht werden, dass BGP-Hijacking ein zunehmendes Problem ist und sie ihre IP-Präfixe durch RPKI schützen müssen.

Für eine erneute Untersuchung müssen zum einen nicht alle Domains mit und ohne dem „www“-Präfix abgefragt werden, sondern nur die ersten 100 000 Ränge, da der Unterschied der IP-Adressen nach Rang 100 000 nur noch gering ist. Zum anderen muss eine andere Liste von ORDNS-Servern gefunden werden, die zuverlässig und schnell antworten und diese müssen viel mehr parallel abgefragt werden, damit die Messung in kürzerer Zeit durchgeführt werden kann.

Dadurch kann die Veränderung der IP-Adressen und auch die veränderlichen Antwortzeiten der ORDNS-Server vermieden werden, das die Messung verfälschen kann. Allerdings wurde gezeigt, dass bei den hinteren Rängen sich die IP-Adressen nicht mehr besonders unterscheiden und dadurch besonders die ersten Ränge gut verteilt abgefragt werden müssen. Bei kleineren Messungen kann das PlanetLab besser verteilt sein, wenn bei ORDNS-Servern nicht alles abgefragt werden kann.

Literatur

[1] V. K. Adhikari, S. Jain, Y. Chen und Z.-L. Zhang, „Vivisecting YouTube: An active measurement study“, in *INFOCOM, 2012 Proceedings IEEE*, 2012, S. 2521–2525.

[2] B. Ager, W. Mühlbauer, G. Smaragdakis und S. Uhlig, „Comparing DNS Resolvers in the Wild“, in *Proceedings of Internet Measurement Conference (IMC '10)*, Melbourne, Australia: ACM, Nov. 2010, S. 15–21.

[3] B. Ager, W. Mülbauer, G. Smaragdakis und S. Uhlig, „Web Content Cartography“, in *Proceedings of Internet Measurement Conference (IMC '11)*, Berlin, Germany: ACM, Nov. 2011, S. 585–600.

[4] R. Atkinson und M. Fanto, *RIPv2 Cryptographic Authentication*, RFC 4822 (Proposed Standard), Internet Engineering Task Force, Feb. 2007.

[5] T. Bates, R. Bush, T. Li und Y. Rekhter, *DNS-based NLRI origin AS verification in BGP*, Internet Draft, Internet Engineering Task Force, Juli 1998.

[6] L. Blunk, M. Karir und C. Labovitz, *Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format*, RFC 6396 (Proposed Standard), Internet Engineering Task Force, Okt. 2011.

[7] P. Boothe, J. Hiebert und R. Bush. (2005). How prevalent is prefix hijacking on the internet?, Adresse: `http://141.223.82.74/research/08/KT_BGP/Presentation/boothe-hijacking.pdf`.

[8] BuiltWith. (2013). CDN Usage Statistics, Adresse: `https://trends.builtwith.com/cdn`.

[9] R. Bush und R. Austein, *The Resource Public Key Infrastructure (RPKI) to Router Protocol*, RFC 6810 (Proposed Standard), Internet Engineering Task Force, Jan. 2013.

[10] K. Butler, T. Farley, P. McDaniel und J. Rexford, „A Survey of BGP Security Issues and Solutions“, *Proceedings of the IEEE*, Bd. 98, Nr. 1, S. 100–122, Jan. 2010.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley und W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280 (Proposed Standard), Updated by RFC 6818, Internet Engineering Task Force, Mai 2008.

[12] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDanial und A. Rubin, „Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing“, in *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS '03)*, 2003.

[13] A. Heffernan, *Protection of BGP Sessions via the TCP MD5 Signature Option*, RFC 2385 (Proposed Standard), Obsoleted by RFC 5925, updated by RFC 6691, Internet Engineering Task Force, Aug. 1998.

[14] R. Hiran, N. Carlsson und P. Gill, „Characterizing Large-scale Routing Anomalies: A Case Study of the China Telecom Incident“, in *Proceedings of the 14th International Conference on Passive and Active Measurement*, Ser. PAM'13, Hong Kong, China: Springer-Verlag, 2013, S. 229–238.

[15] C. Huang, A. Wang, J. Li und K. W. Ross, „Measuring and Evaluating Large-scale CDNs“, in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, Ser. IMC '08, Vouliagmeni, Greece: ACM, 2008, S. 15–29.

[16] C. Huang, A. Wang, J. Li und K. W. Ross, „Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own Red Swoosh“, in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Ser. NOSSDAV '08, Braunschweig, Germany: ACM, 2008, S. 75–80.

[17] G. Huston und R. Bush, „Securing BGP“, *The Internet Protocol Journal*, Bd. 14, Nr. 2, Juni 2001.

[18] S. T. Kent, „Securing the Border Gateway Protocol“, *The Internet Protocol Journal*, Bd. 6, Nr. 3, Sep. 2003.

[19] W. Kumari und K. Sriram, *Recommendation for Not Using AS_SET and AS_CONFED_SET in BGP*, RFC 6472 (Best Current Practice), Internet Engineering Task Force, Dez. 2011.

[20] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide und F. Jahanian, „Internet Inter-domain Traffic“, in *Proceedings of the ACM SIGCOMM 2010 Conference*, Ser. SIGCOMM '10, New Delhi, India: ACM, 2010, S. 75–86.

[21] M. Lepinski und S. Kent, *An Infrastructure to Support Secure Internet Routing*, RFC 6480 (Informational), Internet Engineering Task Force, Feb. 2012.

[22] M. Lepinski und S. Turner, *An Overview of BGPSEC*, Internet Draft, Internet Engineering Task Force, Juli 2013.

[23] W. Lian, E. Rescorla, H. Shacham und S. Savage, „Measuring the practical impact of DNSSEC Deployment“, in *Proceedings of the 22nd USENIX conference on Security*, USENIX Association, 2013, S. 573–588.

[24] J. Liang, J. Jiang, H. Duan, K. Li und J. Wu, *Measuring query latency of top level dns servers*, 2013.

[25] C. Lynn, S. Kent und K. Seo, *X.509 Extensions for IP Addresses and AS Identifiers*, RFC 3779 (Proposed Standard), Internet Engineering Task Force, Juni 2004.

[26] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck und J. Wang, „A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers“, in *Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference*, Ser. ATEC '02, Berkeley, CA, USA: USENIX Association, 2002, S. 229–242.

[27] Matthias Wählisch, Fabian Holler, Thomas C. Schmidt und Jochen H. Schiller, „RTRlib: An Open-Source Library in C for RPKI-based Prefix Origin Validation“, in *Proceedings of USENIX Security Workshop CSET'13*, Berkeley, CA und USA: USENIX Assoc, 2013.

[28] J. Mauch. (2013). Spoofing ASNs (Re: SNMP DDoS: the vulnerability you might not know you have), Adresse: `http://mailman.nanog.org/pipermail/nanog/2013-August/060256.html`.

[29] P. Mockapetris, *Domain names - concepts and facilities*, RFC 1034 (INTERNET STANDARD), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, Internet Engineering Task Force, Nov. 1987.

[30] P. Mockapetris, *Domain names - implementation and specification*, RFC 1035 (INTERNET STANDARD), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, Internet Engineering Task Force, Nov. 1987.

[31] O. Nordström und C. Dovrolis, „Beware of BGP Attacks“, *SIGCOMM Comput. Commun. Rev.*, Bd. 34, Nr. 2, S. 1–8, Apr. 2004.

[32] P. v. Oorschot, T. Wan und E. Kranakis, „On Interdomain Routing Security and Pretty Secure BGP (psBGP)“, *ACM Transactions on Information and System Security*, Bd. 10, Nr. 3, Juli 2007.

[33] G. Peng, „CDN: Content Distribution Network“, *eprint arXiv:cs/0411069*, Nov. 2004.

[34] PerlMonks. (2007), Adresse: `http://www.perlmonks.org/?node_id=623894`.

[35] P. Pillay-Esnault und A. Lindem, *OSPFv3 Graceful Restart*, RFC 5187 (Proposed Standard), Internet Engineering Task Force, Juni 2008.

[36] Y. Rekhter, T. Li und S. Hares, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271 (Draft Standard), Updated by RFCs 6286, 6608, 6793, Internet Engineering Task Force, Jan. 2006.

[37] RIPE NCC News. (2008). YouTube Hijacking: A RIPE NCC RIS case study, Adresse: `http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study`.

[38] K. Schomp, T. Callahan, M. Rabinovich und M. Allman, „On Measuring the Client-side DNS Infrastructure“, in *Proceedings of the 2013 Conference on Internet Measurement Conference*, Ser. IMC '13, Barcelona, Spain: ACM, 2013, S. 77–90.

[39] A.-J. Su, D. R. Choffnes, A. Kuzmanovic und F. E. Bustamante, „Drafting Behind Akamai (Travelocity-based Detouring)“, in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Ser. SIGCOMM '06, Pisa, Italy: ACM, 2006, S. 435–446.

[40] A.-J. Su, D. R. Choffnes, A. Kuzmanovic und F. E. Bustamante, „Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections“, *IEEE/ACM Trans. Netw.*, Bd. 17, Nr. 6, S. 1752–1765, Dez. 2009.

[41] J. Touch, A. Mankin und R. Bonica, *The TCP Authentication Option*, RFC 5925 (Proposed Standard), Internet Engineering Task Force, Juni 2010.

[42] S. Triukose, Z. Wen und M. Rabinovich, „Measuring a Commercial Content Delivery Network“, in *Proceedings of the 20th International Conference on World Wide Web*, Ser. WWW '11, Hyderabad, India: ACM, 2011, S. 467–476.

[43] Q. Vohra und E. Chen, *BGP Support for Four-Octet Autonomous System (AS) Number Space*, RFC 6793 (Proposed Standard), Internet Engineering Task Force, Dez. 2012.

[44] M. Wählisch, O. Maennel und T. C. Schmidt, „Towards Detecting BGP Route Hijacking Using the RPKI“, in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Ser. SIGCOMM ’12, Helsinki, Finland: ACM, 2012, S. 103–104.

[45] R. White, „Securing BGP Through Secure Origin BGP“, *The Internet Protocol Journal*, Bd. 6, Nr. 3, Sep. 2003.

[46] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu und L. Zhang, „An Analysis of BGP Multiple Origin AS (MOAS) Conflicts“, in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, Ser. IMW ’01, San Francisco, California, USA: ACM, 2001, S. 31–35.

Abkürzungsverzeichnis

AFRNIC	African Network Information Centre
APNIC	Asia-Pacific Network Information Centre
ARIN	American Registry for Internet Numbers
AS	Autonomes System
ASN	Autonome System Nummer
BGP	Border Gateway Protocol
CA	Certificate Authority
CDN	Content Delivery Network
CPE	Customer Premises Equipment
CSV	Comma-separated values
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EE	End-Entity-Zertifikat
EGP	Exterior Gateway Protocols
FIB	Forwarding Information Base
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
IGP	Interior Gateway Protocols
IP	Internet Protocol (IPv4: Version 4, IPv6: Version 6)
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LACNIC	Latin American and Caribbean Internet Addresses Registry
LIR	Local Internet Registries
MOAS	Multiple Origin AS

NLRI	Network Layer Reachability Information
ORDNS	Open Recursive DNS-Server
PIB	Policy Information Base
PKI	Public Key Infrastructure
RC	Ressource Zertifikat
RIB	Routing Information Base
RIPE	Réseaux IP Européens Network Coordination Centre (RIPE NNC)
RIR	Regional Internet Registry
ROA	Route Origin Autorization Object
RPKI	Resource Public Key Infrastructure
RTR	RPKI to Router Protocol
SIDR	secure Inter-Domain Routing
TCP	Transmission Control Protocol
TLD	Top Level Domain
TTL	Time To Live
URL	Uniform Resource Locator