(Secure) IPv6

Elias Pichottka
Universität Potsdam

8. Juli 2020

Zusammenfassung

In dieser Arbeit wird das Internet Protokoll in der Version 6 (IPv6) vorgestellt, unter Betrachtung von sicherheitsrelevanten Aspekten.

1 Einleitung

Diese Arbeit soll einen generellen Überblick zum Internet Protokoll in der Version 6 (IPv6) bieten und zusätzlich auf sicherheitsrelevante Erweiterungen und Sicherheitslücken eingehen. Im Abschnitt 2 wird die Nutzung des IPv6 durch das Internet Protokoll Version 4 (IPv4) Adressproblem motiviert und eine alternative Lösung, Network Adress Translation (NAT), aufgezeigt. Der Abschnitt 3 beschäftigt sich sich mit dem IPv6 Header und dessen mögliche Erweiterungen, durch Extension Header. Ein Vergleich zum Vorgänger Protokoll, das Internet Protokoll in der Version 4 (IPv4), wird im Abschnitt 4 behandelt. Auf die Protokoll Suite IP Security (IPsec) wird im Abschnitt 5 eingegangen. Im Abschnitt 6 wird das Internet Control Message Protocol (ICMPv6) beleuchtet. Der Abschnitt 7 geht auf zwei Angriffe ein, welche über das IPv6 ausgeführt werden können. Zum Ende der Arbeit, in Abschnitt 8, wird eine Client/Server Anwendung unter Verwendung des IPv6 vorgestellt, gefolgt von einem persönlichen Fazit, in Abschnitt 9.

2 Motivation

Die Anzahl der Haushalte mit einer Internetverbindung steigt stetig, dies lässt durch Abbildung 1 [1] leicht nachvollziehen. Abbildung 1 stellt die relative Anzahl an Haushalten dar, welche eine Internet Verbindung (helles Blau) haben und welche einen PC besitzen (dunkles Blau). Hierbei repräsentiert die Y-Achse den relativen Anteil der Haushalte in Prozent und die X-Achse das Jahr der Messung, von 2005 bis 2019. Die jetzige Welbevölkerung beträgt ungefähr $7.8 * 10^9$ Menschen [3] und oft gibt es mehrere internetfähige Geräte pro Haushalt und sogar pro Person. Wenn man nun die Größe des IPv4 Adressraums mit $2^{32} \approx 4.3 * 10^9$ Adressen betrachtet, wird es klar das es nicht genügend

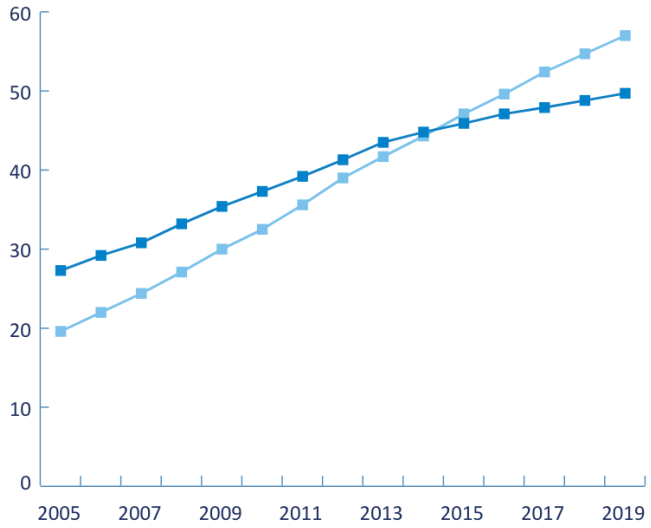

Abbildung 1: Haushalte mit Internetverbindung (helles Blau) und PC (dunkles Blau)

IPv4 Adressen für weltweit alle Geräte gibt, dieses Problem wird auch Adressproblem genannt. Ein Lösungsansatz für das Adressproblem ist die Network Adress Translation (NAT). Durch NAT werden weniger öffentliche beziehungsweise globale IPv4 Adressen genutzt, in dem mehrere Geräte in einem privaten Netzwerk, private unterschiedliche IP Adressen erhalten, aber insgesamt nur eine öffentlich IP Adresse zugeteilt wird. Dieses Konzept lässt sich auch zusätzlich auf die Ebene des Internet Service Providers (ISP) erweitern, in dem dieser ebenfalls ein privates Netzwerk für all seine Kunden führt. In Abbildung 2 ist diese mehrstufige NAT dargestellt, wobei die roten Boxen für NAT auf der Ebene des Kunden und die Blaue Box für NAT auf der Ebene des ISP steht. Hierbei ist es jedoch nachteilig, dass die einzelnen Router einen erhöhten Arbeitsaufwand durch die Übersetzung der Adressen haben, der Netzwerkaufbau komplizierter wird und in der Regel auch der Stromverbrauch ansteigt. Eine andere Alternative, zur Lösung des Adressproblems, die deutich besser skaliert ist IPv6. Durch die Verwendung von 128-bit Adressen ist das IPv6 in der Lage $2^{128} \approx 3.4 * 10^{38}$ Adressen darzustellen. In den letzten Jahren ist die Verwendung des IPv6 stark gestiegen, dies kann man Anhand der Nutzungsstatistiken für die Suchmaschine Google, in Abbildung 3 [6], nachvollziehen. Die X-Achse der Abbildung 3 steht hierbei für das Datum der Messung und die Y-Achse für den relativen Anteil der IPv6 Google Nutzung, im Vergleich zur gesamten Google Nutzung. Seit Anfang 2015 steigt die IPv6 Google Nutzung um circa 5% pro Jahr, heutzutage werden ungefähr ein Drittel aller Google Zugriffe über das IPv6 getätigt.

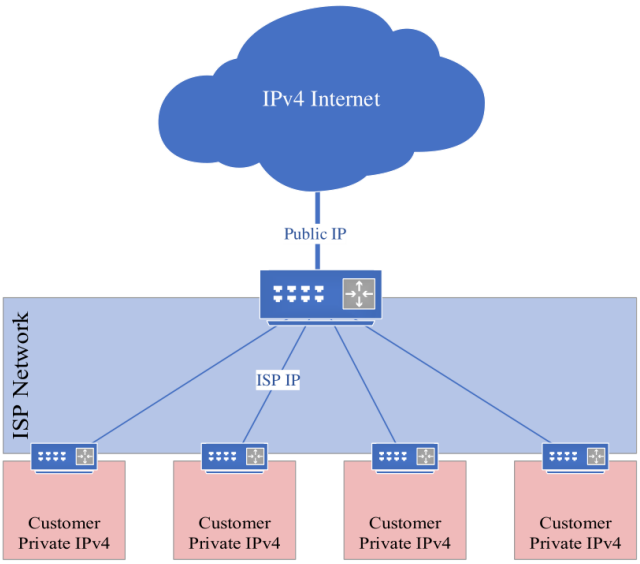

Abbildung 2: Network Adress Translation (NAT)

3 IPv6 Standard

Der IPv6 Standard wurde durch die Internet Engineering Task Force (IETF) erarbeitet. Die IETF ist eine internationale Vereinigung von Freiwilligen zur Standardisierung des Internets, welche in Arbeitsgruppen organisiert ist. Die Früchte der Arbeit der IETF sind technische Dokumente, sogenannte Requests for Comments (RFC's). RFC's werden in folgende Typen eingeteilt:

- Informational
- Experimental
- Standards Track
- Historic

In dieser Arbeit werden hauptsächlich RFC's die einen Internet Standard bilden (Typ: Standard Track) und vereinzelt auch überholte Standards (Typ: Historic) betrachtet. Das IPv6 wurde hauptsächlich durch die Arbeitsgruppe IP next generation (IPng) erarbeitet. Der erste Standard wurde im Jahre 1995 als RFC 1883 veröffentlicht, im Jahr 1998 durch RFC 2460 ersetzt, und später, 2017, durch den aktuellen Standard, RFC 8200.

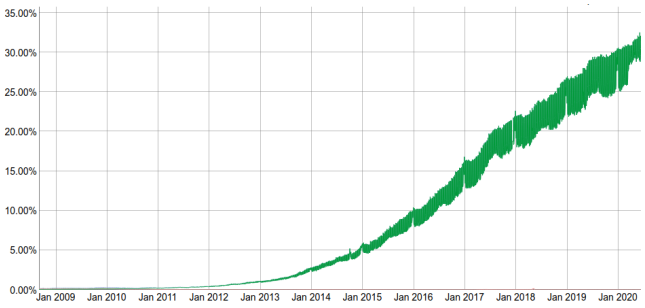

Abbildung 3: Google Nutzung über IPv6

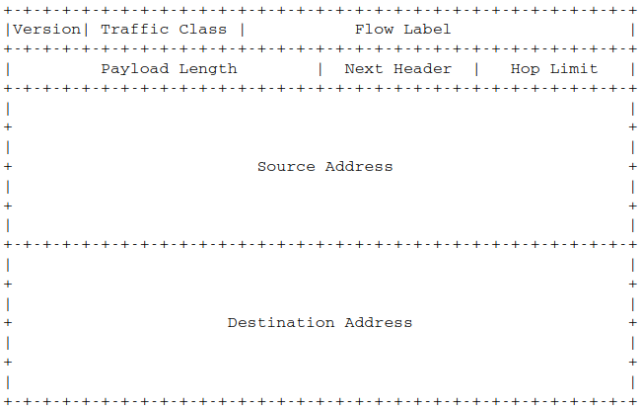

Abbildung 4: IPv6 Header nach RFC 8200

In Abbildung 4 ist der Aufbau des IPv6 Headers, nach RFC 8200 [8], sichtbar. Nachfolgend wird auf die Größe und Verwendung der einzelnen Header-Felder eingegangen.

Version (4-bit)	belegt durch Internet Protokoll Version, sprich 6
Traffic Class (8-bit)	wird für Packet-Priorität genutzt
Flow Label (20-bit)	kennzeichnet zusammenhängende Pakete
Payload Length (16-bit)	Länge der folgendend Daten inkl. Extension Header in Byte
Next Header (8-bit)	gibt Typ des nächsten Extension Headers vor
Hop Limit (8-bit)	gibt maximale Anzahl an Weiterleitungen an, wird an jedem Knoten dekrementiert
Source Adress (128-bit)	Adresse des Paket-Absenders
Destination Adress (128-bit)	Ziel Adresse für Paket

Wie bereits erwähnt wird im Next Header Feld der Typ des nächsten Extension Headers angegeben. Ein Extension Header ist hierbei eine Header Erweiterung die am Ende des vorherigen Headers angefügt werden kann und dessen Funktionaltität vergrößern kann.

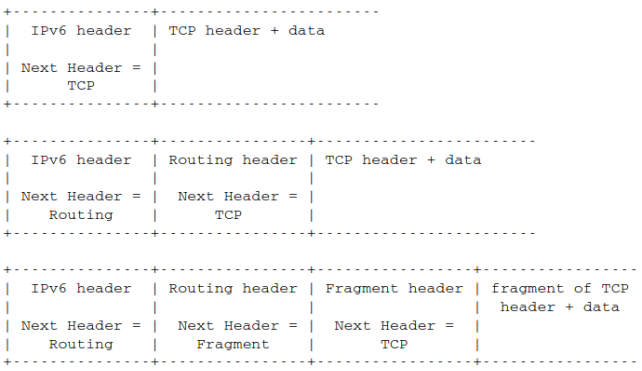

Abbildung 5: IPv6 Header Extensions nach RFC 8200

In Abbildung 5 [8] ist die Verkettung von IPv6 Header und Extension Header verdeutlicht. Im oberen Teil der Abbildung 5 ist ein minimales Beispiel ohne Extension Header aufgezeigt, hierbei enthält das Next Header Feld des IPv6 Header TCP, ausgeschrieben Transmission Control Protocol, dadurch wird signalisiert das ein TCP-Header sowie Nutzdaten folgen. In der Mitte von Abbildung 5 ist die Erweiterung des minimalen Beispiels durch einen Routing Header als Extension Header zu sehen. Im unteren Teil der Abbildung 5 wird das vorhergehende Beispiel noch zusätzlich durch einen Fragment Header erweitert und es handelt sich dementsprechend um ein Fragment/Teil von Nutzdaten das versendet wird.

4 Vergleich zum IPv4

Das Internet Protocol Version 4 (IPv4) besitzt eine komplexere Header Struktur als der IPv6 Header, da ein großer Teil optionaler Funktionaltiät auf Extension Header ausgelagert wurde. Der IPv4 Header, nach RFC 791 [12], ist in Abbildung 6 zu sehen. Im Folgenden werden IPv4 Header Felder die in den IPv6 Header:

- übernommen wurden durch '=',
- welche unter anderem Namen übernommen wurden durch 'r',
- und verschwundene Felder durch 'x'

gekennzeichnet. Falls es sich um eine Umbennung handelt wird der neue Name des IPv6 Feldes erwähnt, andernfalls wird Auskunft über das IPv4 Feld gegeben.

Version (4-bit) = nur andere Belegung

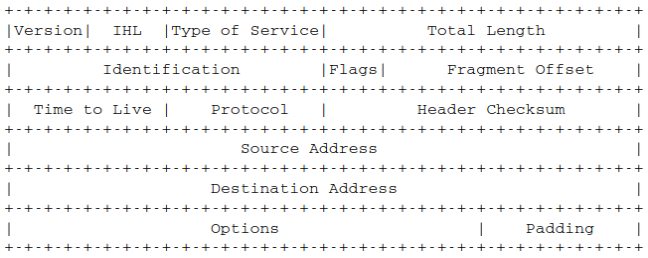

Abbildung 6: IPv4 Header nach RFC 791

IHL (4-bit)	x	Internet Header Length, obsolet durch IPv6 Header mit fester Größe
Type of Service (8-bit)	r	IPv6 Traffic Class
Total Length (16-bit)	r	IPv6 Payload Length
Identification (16-bit)	x	Nummerierung von Paketen
Flags (3-bit)	x	Information zu Fragmentierung
Fragment Offset (13-bit)	x	gibt Position des Fragments in Dateneinheit an
Time to Live (8-bit)	r	IPv6 Hop Limit
Protocol (8-bit)	r	gibt Protokoll für Datensektion an, IPv6 Next Header
Header Checksum (16-bit)	x	sichert Korrektheit des IPv4 Headers
Source Adress (32-bit)	=	lediglich 4-fache Größe
Destination Adress (32-bit)	=	lediglich 4-fache Größe
Options + Padding	x	(variabel) Routing, Debugging oder Statistische Information aufgefüllt mit Nullen

Neben den genannten Änderungen, ist auch die Schreibweise von IPv4 Adressen unterschiedlich zur Schreibweise von IPv6 Adressen. IPv4 Adressen sind 32-bit lang und können daher relativ kompakt durch 4 Dezimalzahlen dargestellt werden. Als Trennzeichen zwischen den 4 Dezimalzahlen werden jeweils einfache Punkte verwendet. So kann durch zum Beispiel: 127.0.0.1 die lokale IPv4 Loopback-Adresse beschreiben werden. Im Gegensatz dazu sind IPv6 Adressen 128-bit lang, dies führt zu einer im Regelfall längeren Schreibweise. Es hat sich etabliert eine Hexadezimaldarstellung aus 8 Zahlen zu verwenden, wobei das Trennzeichen ein Doppelpunkt ist. Um trotz größerer Datenmenge die Darstellung ohne Datenverlust weiter zu kürzen, ist es erlaubt führende Nullen und Blöcke von Nullen zusammenzufassen. Zusätzlich können die letzten 32 bits der Adresse IPv4 konform dargestellt werden. Diese syntaktische Vielfalt für semantisch gleiche Adressen macht es jedoch schwieriger Adressen zu vergleichen. Im nachfolgenden Beispiel ist eine IPv6 Adresse auf 4 verschiedene Arten niedergeschrieben:

- 0000 : 0000 : 0000 : 0000 : 0000 : $ffff$: 7f00 : 0001
- :: $ffff$: 7f00 : 0001
- :: $ffff$: 7f00 : 1
- :: $ffff$: 127.0.0.1

5 IPsec

Internet Protocol Security (IPsec) ist eine Protokoll Suite/Sammlung bestehend aus:

- Authentication Header (AH)
- Encapsulating Securtiy Payload (ESP)

IPsec wird zur gesicherten Kommunikation durch Verschlüsselung genutzt und findet zum Beispiel bei Virtual Private Networks (VPN's) Anwendung. Der jetzige IPsec Standard ist im RFC 4301 [11] verfasst und ersetzt damit RFC 2401. In Abbildung 7 ist der Authentication Header gezeigt, er wird genutzt um

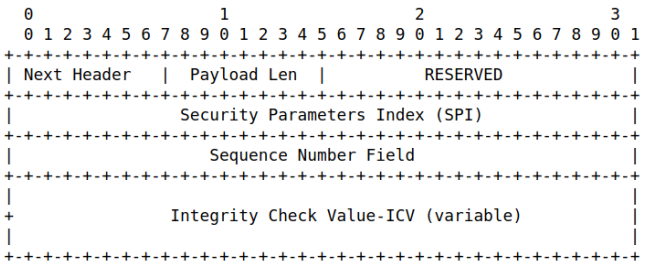

Abbildung 7: Authentication Header nach RFC 4301

die Quelle des Pakets (den Absender) zu authentifizieren beziehungsweise zu bestätigen. Durch den AH kann die Datenintergrität von Header und Payload gewährleistet werden. Um ein AH zu nutzen muss der Next Header Wert 51 verwendet werden. Im Folgenden sind die Felder des AH kurz beschrieben.

Next Header (8-bit)	Typ des nächsten Payloads
Payload Length (8-bit)	Länge des AH (in 4-byte Wörtern)
RESERVED (16-bit)	reserviert für Erweiterte Funktionalität
Security Parameter Index (32-bit)	Tag zu Unterscheidung verschiedener Datenströme
Sequence Number Field (32-bit)	Counter wird Paketweise Erhöht für eine Datenstrom
ty Check Value (multiple of 32-bit)	aus Header und Payload bestimmter Wert (Datenintegrität)

Im Gegensatz zum AH kann der Encapsulating Security Payload (ESP) nicht nur zur Authentifizierung eines Datenpakets genutzt werden, sondern auch zur Verschlüsselung eines Datenpakets. Abbildung 8 zeigt den Aufbau des ESP Hea-

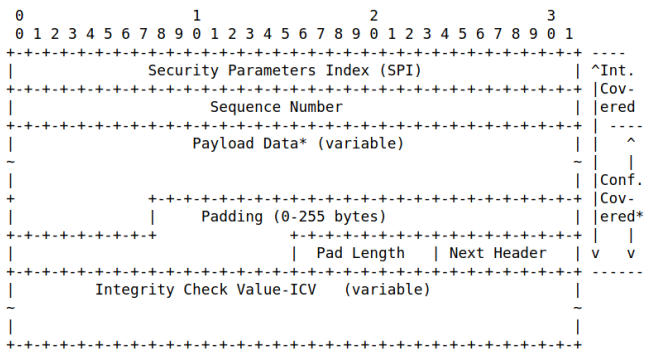

Abbildung 8: Encapsulating Security Payload Header nach RFC 4301

der. Um den ESP Header zu verwenden ist es notwendig, im vorherigen Header, den Next Header Wert 50 zu setzen. Im Folgenden werden die Header Felder des ESP kurz beschreiben:

Security Parameter Index (32-bit)	Tag zu Unterscheidung verschiedener Datenströme
Sequence Number (32-bit)	Counter wird Paketweise Erhöht für eine Datenstrom
Payload Data (multiple 1-byte)	variable Länge enthält verschlüsselte Daten
Padding (0-255 bytes)	sinnvoll für Verschlüsselungsalgorithmen die gewisse Datenlänge erwarten
Pad Length (1-byte)	Länge vom Padding
Next Header (8-bit)	Typ vom folgenden Extension Header
Integrity Check Value (variable)	berrechnet aus ESP Header und Payload

6 ICMPv6

Das Internet Control Message Protocol Version 6 (ICMPv6) wird für Fehler- und Statusmeldungen genutzt. Der aktuelle ICMPv6 Standard ist im RFC 4443 [13] gefasst und um das ICMPv6 zu nutzen ist der Next Header Wert 58 notwendig.

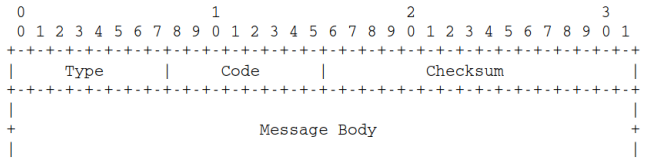

Abbildung 9: ICMPv6 Header nach RFC 4443

Das generelle ICMPv6 Header Format ist in Abbildung 9 gezeigt. Die einzelnen Felder des Headers werden im Folgenden kurz beschrieben:

Type der Typ der Nachricht

Code unterschiedliche Nutzung je nach Nachrichtentyp

Checksum benutzt zum Erkennen von Fehlern im Header sowie Message-Body

Message Body enthält Nachricht entsprechend Typ

Typ 1 bis 127 wird für Fehlermeldungen genutzt, die folgenden Typen sind im RFC 4443 definiert:

1 Destination Unreachable

2 Packet Too Big

3 Time Exceeded

4 Parameter Problem

100 Private experimentation

101 Private experimentation

127 Reserved for expansion of ICMPv6 error messages

Typ 128 bis 255 wird für Stausmeldungen genutzt, die folgenden Typen sind im RFC 4443 definiert:

128 Echo Request

129 Echo Reply

200 Private experimentation

201 Private experimentation

255 Reserved for expansion of ICMPv6 informational messages

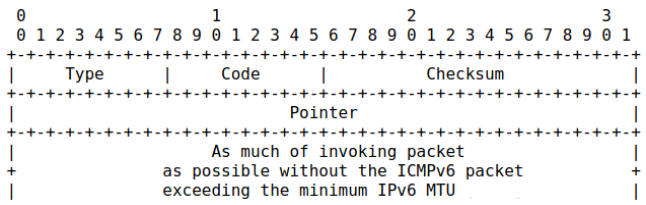

Abbildung 10: ICMPv6 Typ 4 Header

In Abbildung 10 ist der Header für eine ICMPv6 Meldung vom Typ 4 (Parameter Problem) gezeigt. Im Folgenden werden die für den ICMPv6 Typ 4 Header spezifischen Felder beschrieben:

Code 0 Error im Header

1 unbekannter next Header Typ

2 unbekannte IPv6 Option

Pointer zeigt auf Stelle an dem Fehler aufgetreten ist

Message Body möglichst viel des ursprünglichen Pakets ohne größer als MTU zu sein

Der Begriff MTU steht in der obigen Beschreibung für Maximal Transmission Unit (deutsch: maximale Übertragungseinheit).

7 IPv6 Angriffe

Dieser Abschnitt wird sich mit dem

- Routing Header Typ 0 Angriff und
- dem Smurf Amplifier Angriff

befassen. Der Routing Header von Typ 0 kann eine beliebige Anzahl an Wei-

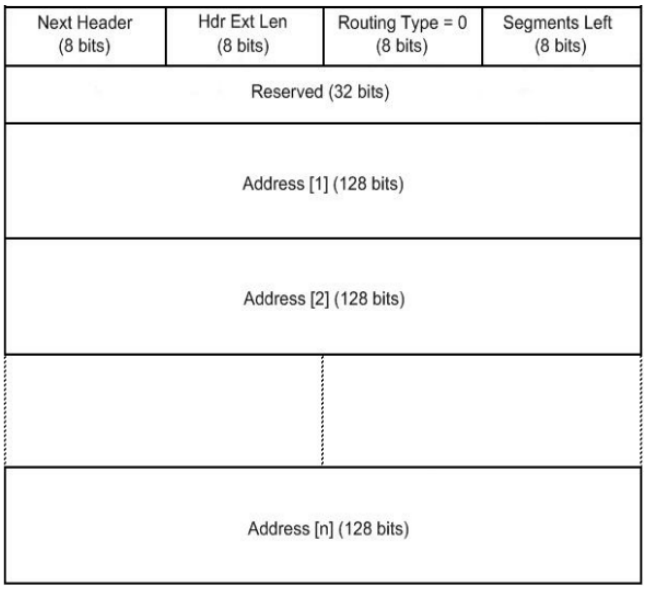

Abbildung 11: Routing Header von Typ 0

terleitungsadressen enthalten, der generelle Aufbau ist in Abbildung 11 [14] gezeigt. Um den Routing Header von Typ 0 zu nutzen muss der Next Header Wert 43 spezifiert sein. Da die Möglichkeit besteht Weiterleitungsadressen mehrfach anzugeben, kann man durch die Wiederholung von 2 Adressen erreichen das Pakete zwischen 2 Routern pendeln. Dies kann dazur führen das die Router

keine weiteren Pakete mehr verarbeiten können und somit in einem Denial of Service (DoS) enden. Eine Gegenmaßnahme für diesen Angriff bietet der Ingress Filter. Ein Ingressfilter blockiert bestimmte Pakete (e.g. Routing Header Typ 0 Pakete). Die Smurf Amplifier Attacke ist bereits durch das IPv4 bekannt. Bei dieser Attacke sendet der Angreifer modifizierte Pakete mit der Quelladresse des Opfers und als Ziel Adresse eine Multicast-Adresse, welche mehrere Zieladressen zusammenfasst. Im gefälschten Paket muss eine von den Zieladressen nicht unterstützte Option gewählt sein und die höchsten 2 bits des Options-Typ müssen 10 gesetzt sein. Erhalten die Empfänger nun die gefälschten Pakete, so sendet jeder Empfänger jeweils eine ICMPv6 Typ 4 (Parameter Problem) Fehlermeldung, da der Options-Typ nicht unterstützt wird. Das kann in einem DoS auf Seite des Opfers führen, wenn zu viele Fehlermeldungen ankommen. Das

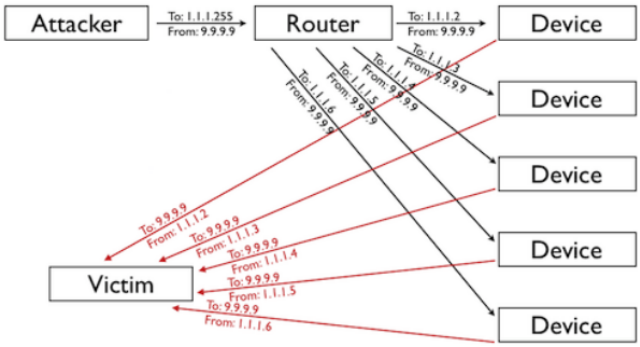

Abbildung 12: Visualisierung Smurf Amplifier Attacke

beschriebene Konzept des Smurf Amplifier Angriffs kann auch Anhand von Abbildung 12 [15] nachvollzogen werden, wobei die genutzten Adressen dem IPv4 entsprechen, das Prinzip ist jedoch analog.

8 Client Server Anwendung

In diesem Abschnitt wird eine minimale Client/Server Anwendung, in Python, unter Verwendung des socket Moduls beschrieben.

```
1  import socket
2
3  reciever_ip = "::1"
4  port = 1337
5  message = "Hello, World!"
6
7  client_socket = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, 0)
8  client_socket.connect((reciever_ip, port, 0, 0))
9  client_socket.send(message)
10
11 print "Client: \"{}\" has been sent to {}".format(message, reciever_ip)
```

Listing 1: client.py

Listing 1 zeigt hierbei den Quelltext der Client Anwendung. Das Client Programm eröffnet zuerst einen socket, verbindet sich dann mit dem Empfänger, dem Server und sendet im Anschluss eine 13 Byte lange Nachricht, “Hello World!”. Die Eingaben der genutzten Funktion sind im Folgenden aufgelistet:

socket(3) Internet Protokoll, Protokoll Klasse für Datenübertragung, Index Datenübertragungsprotokoll

connect(1) IP Adresse, Port, Flow Label, Scope Id

send(1) Daten die gesendet werden

```
1  import socket
2
3  server_ip = "::1"
4  port = 1337
5
6  server_socket = socket.socket(socket.AF_INET6,socket.SOCK_STREAM,0)
7  server_socket.bind((server_ip,port,0,0))
8  server_socket.listen(1)
9  connection, sender = server_socket.accept()
10 message = connection.recv(13)
11
12 print "Server: received \"{}\" from {}".format(message,sender[0],sender
       [1])
```

Listing 2: server.py

Listing 2 zeigt den Quelltext der Server Anwendung. Das Server Programm eröffnet ebenfalls ein socket, lauscht dann auf diesem beziehungsweise wartet auf eine Verbindung, wurde diese Verbindung hergestellt so nimmt der Server 13 Byte vom Client entgegen, was in diesem Beispiel der “Hello World!” Nachricht entspricht. Durch die Client Anwendung noch nicht verwendete Funktionen des Servers werden im Folgenden beschrieben:

bind(1) definiert Adresse, Port, Flow Label, Scope Id mit denen gelauscht werden soll

listen(1) max. Anzahl der Verbindungen

accept() stellt Verbindung mit Client her

recv(1) gibt an wieviel Bytes entgegenommen werden

Als sinnvolle Erweiterung der Server Anwendung könnte nun noch ein Dual-Stack Server implementiert werden. Ein Dual-Stack Server wartet in einer Schleife auf entweder IPv4 oder IPv6 Verbindungen und ist über IPv4 sowie IPv6 adressierbar.

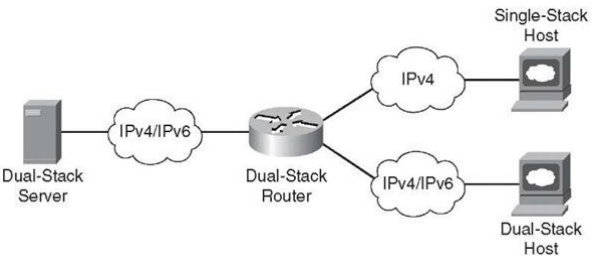

Abbildung 13: Visualisierung Dual-Stack Server

In Abbildung 13 [2] ist dieses Konzept veranschaulicht.

9 Fazit

Diese Arbeit sollte gezeigt haben, das IPv6 hat durch Extension Header viel zu bieten, auch hinsichtlich der Übertragungssicherheit. Weiterhin lässt es sich auch relativ bequem in Anwendungen nutzen, jedoch könnte der Umstieg, von IPv4 auf IPv6, schneller von statten gehen, wenn Bremsen wie zum Beispiel die NAT der ISP gelöst werden würden.

Literatur

[1] "Measuring digital development - itu," 2019.

[2] K. Mohbey, "Future internet plan using ipv6 protocol," *International Journal of Scientific and Engineering Research*, vol. 3, 2012.

[3] "Real time world statistics - worldometer," 2020.

[4] F. Seele, "Neue Möglichkeiten für Lastverteilung in Netzwerken," Master's thesis, University of Potsdam, 2015.

[5] C. Frieben, "IPv6 im Wandel - Analyse der IPv6-Standardisierung unter Sicherheitsaspekten," Master's thesis, University of Potsdam, 2014.

[6] "Ipv6 statistics - google," 2020.

[7] J. D. Sangam Racherla, *IPv6 Introduction and Configuration.* IBM Redbooks, 2012.

[8] R. H. S. Deering, "Internet protocol, version 6 (ipv6) specification," RFC 8200, RFC Editor, 2017.

[9] S. Kent, "Ip encapsulating security payload (esp)," RFC 4303, RFC Editor, 2005.

[10] S. Kent, "Ip authentication header," RFC 4302, RFC Editor, 2005.

[11] S. Kent, "Security architecture for the internet protocol," RFC 4301, RFC Editor, 2005.

[12] J. Postel, "Darpa internet program protocol specification," RFC 791, RFC Editor, 1981.

[13] S. D. A. Conta, "Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification," RFC 4443, RFC Editor, 2006.

[14] "Cisco security threat and vulnerability intelligence," 2014.

[15] "Visualization for smurf amplifier attack - cloudflare," 2019.