INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN


**Bachelorarbeit**

# Evaluierung ausgewählter Methoden zum Scannen des IPv6 Adressbereiches im Münchner Wissenschaftsnetz

Alexander Opalic

Draft vom 12. Juni 2019

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN


**Bachelorarbeit**

# Evaluierung ausgewählter Methoden zum Scannen des IPv6 Adressbereiches im Münchner Wissenschaftsnetz

Alexander Opalic

| | |
|---|---|
| Aufgabensteller: | Prof. Dr. Helmut Reiser |
| Betreuer: | Tobias Appel |
| Abgabetermin: | 13. Juni 2019 |

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Geretsried, den 12. Juni 2019

.....
*(Unterschrift des Kandidaten)*

## Kurzfassung

Die vorliegende Bachelorarbeit behandelt das Thema, anhand ausgewählter Methoden IPv6 Netzwerkscanner miteinander zu vergleichen. Ziel dieser Arbeit ist es zunächst Kriterien für den Vergleich von IPv6 Netzwerkscannern zu definieren. Mithilfe der erstellten Kriterien werden die Netzwerkscanner miteinander verglichen. Auf Basis der dadurch gewonnen Erkenntnisse wird die Praktikabilität von Netzwerkscans an einem IPv6 Netz überprüft. Alle Tests und Vergleiche beziehen sich dabei auf das Münchner Wissensschafts Netz. Die gewonnen Erkenntnisse können aber auf andere Netzwerke übertragen werden. Anhand vier Kriterien, die im Rahmen dieser Arbeit definiert wurden, sind die Netzwerkscanner *Nmap*, *Zmapv6*, *Alive6*, *Scan6* und *Chiron* miteinander verglichen worden. Dabei spielte die Fähigkeit Hosts in einem Netzwerk zu erkennen und den Zustand eines Ports zu bestimmen das wichtigste Kriterium da. Als ein weiteres Kriterien wurde untersucht, welche unterschiedlichen Features der ausgewählte Netzwerkscanner unterstützt. Das letzte Kriterium war die Dokumentation des jeweiligen Netzwerkscanner. Anhand aufgenommener Testdaten, wurde hochgerechnet, wie lange die Scanner brauchen würden, um festgelegte Adressengrößen zu untersuchen. Mit dem Ergebnis der Arbeit kann aufgezeigt werden wie heutige Netzwerkscanner den IPv6 Adressraum eines großen Netzes untersuchen. Darüber hinaus kann in dieser Arbeit geklärt werden ob Netzwerkscanner in IPv6 allein reichen, um sich einen Überblick über angeschlossene Geräte zu machen.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Heutzutage ist das Internet nicht mehr aus unserem Alltag wegzudenken. Oft ist der erste Blick eines Menschen in der modernen westlichen Welt, auf sein Smartphone, nachdem er seinen Alarm ausgeschaltet hat. So überprüft er die neusten Nachrichten oder seine Social Media Aktivitäten.

Was dem Laien dabei aber nicht bewusst ist, das sein Smartphone in der Regel heute noch IPv4 benutzt, um mit der Außenwelt kommunizieren zu können. Dabei steht IPv4 für Internet Protokoll Version 4. Bildlich gesprochen kann das Smartphone durch IPv4 und einer Handvoll anderer Protokolle mit der Außenwelt kommunizieren.

Dabei dient IPv4 als eine Art Adresse an dem andere Teilnehmer durch ein kompliziertes Verfahren Informationen zu diesem hinschicken können. Das Smartphone oder der Computer, oder heutzutage sogar eine Kaffeemaschine befindet sich in einem lokalen Netzwerk. In diesem Netz kommunizieren die Teilnehmer mithilfe von Routern mit anderen extern liegenden Netzwerken.

Bei größeren Netzwerken von Universitäten oder Betrieben, ist es außerdem wichtig, einen Überblick über sein Netzwerk zu haben. So werden Geräte inventarisiert, das heißt es wird genau ermittelt wie viele Router und Computer und weitere Geräte, in dem eigenen Netzwerk sind. Außerdem ist es wichtig, sein eigenes Netzwerk vor externen Angriffen schützen zu können. Um einen Überblick über mögliche Sicherheitsschwachstellen zu haben werden Netzwerkscanner benutzt.

Dabei stellen diese ein wichtiges Werkzeug da, mit dem es Betreibern des Netzes möglich ist, sich einen aktuellen Überblick über den Zustand ihres Netzwerkes zu verschaffen. Dies geschieht sehr vereinfacht ausgedrückt in zwei Schritten. Zuerst wird jede einzelne Adresse, des angeschlossenen Gerätes, des zu untersuchten Netzes entdeckt und dann anschließend jeder Host mit einem Netzwerkscanner genauer untersucht.

Mithilfe von einem Vulnerabilty Scan (engl: Schwachstellenscan) wird der zu untersuchte Host auf potenzielle Angriffsziele untersucht. Aber durch die begrenzte Anzahl von IPv4 Adressen, und die immer größer werdende Zahl an Endgeräten wurde deshalb vor über zwanzig Jahren das Internet Protokoll Version 6 (IPv6) entwickelt.

IPv6 hat einen möglichen Adressraum von 128 Bit im Vergleich zu IPv4 mit 32 Bit. IPv4 wurde in den frühen 70ern entwickelt, zu dieser Zeit gab es keine Smartphones oder Internet fähige Drucker und die Anzahl an verbunden Rechnern war auch sehr begrenzt. Gegen Anfang waren es nur Universitäten, die miteinander kommunizierten. Es war nicht absehbar,

wie viele Endgeräte eines Tages im Internet einmal vernetzt sein würden. IPv6 wird also im Laufe der Zeit immer wichtiger, ein Indikator für die aktuelle Verwendung von IPv6 liefert Google. Auf der Seite [goo18] sieht man, das die Suchanfrage über IPv6 seit Januar 2009 um ca. 23 % gestiegen ist. Auch RIPE welches offiziell dafür verantwortlich ist den Adressraum in IPv4 oder IPv6 an Kunden aufzuteilen hat zum Erstellen dieser Arbeit im Jahre 2019 offiziell vermittelt, dass der gesamte noch frei adressierbare IPv4 Raum nächstes Jahr im Februar 2020 wahrscheinlich ablaufen würde [Ped19].

Es wird also in Zukunft immer bedeutender das Administratoren sich auch in einem IPv6 Netzwerk sich einen Überblick über ihr Netz machen können. Durch den großen Adressbereich von IPv6 im Vergleich zu IPv4 ist es nicht möglich, das ganze Netzwerk mit einer simplen brute Force Anfrage zu scannen. Laut [BHFV18] würde es mit aktuellen Netzwerkscanner $7.532 \cdot 10^{23}$ Jahre dauern, um den gesamten Adressraum von IPv6 untersuchen zu können.

Die vorliegende Bachelorarbeit beschäftigt sich deshalb anhand ausgewählte Netzwerkscanner ob und wie diese sich einen Überblick über ein Netzwerk verschaffen können. Auch das Leibniz-Rechenzentrum (LRZ) führt regelmäßig Netzwerkscans in Münchner Wissenschaftsnetz (MWN) durch um sich einen Überblick über angeschlossene Geräte zu machen. Zurzeit werden diese Scans aber nur für IPv4 Netze gemacht, welche mit einem vertretbaren Zeitaufwand noch möglich sind.

Doch für IPv6 Netze, welche in Zukunft immer mehr an Relevanz gewinnen werden, ist noch kein regelmäßiger Netzwerkscan vorhanden. Hier setzt diese Arbeit an, indem sie in einem gegebenen Netzwerk ausgewählte Methoden zum Scannen des IPv6 Adressbereiches untersuchen wird.

## 1.2 Ziele der Arbeit

Ziel dieser Arbeit ist es aktuelle Netzwerkscanner in einem IPv6 Netz aufgrund festgelegter Kriterien miteinander zu vergleichen. Aufgrund der gewonnen Messdaten und Erkenntnisse des Vergleichs soll darüber hinaus aufgezeigt werden wie praktikabel Scans in einem großen IPv6 Adressraum sein könnten. Folgende Forschungsfragen sollen in dieser Arbeit geklärt werden:

1. Welche Kritieren können benutzt werden um Netzwerkscanner miteinander vergleichen zu können?
2. Wie unterscheiden sich die unterschiedlichen Netzwerkscanner anhand der festgelegten Kriterien?
3. Ist es möglich mit ausgewählten Netzwerkscannern den IPv6 Adressraum effizient zu untersuchen?

## 1.3 Struktur der Arbeit

Die Bachelorarbeit lässt sich in folgende Bereiche untergliedern. Ausgehend von der Einleitung werden in Kapitel 2 und 3 Grundlagen gelegt, die für das Verständnis dieser Arbeit benötigt werden, um das Scannen von IPv6 Adressen besser verstehen zu können.

Bei den Grundlagen handelt es sich um alle theoretischen benötigten Bereiche der auftreten Protokolle, bei einem Netzwerksan für diese Arbeit und insbesondere um die Kenntnisse für das Scannen eines IPv6 Netzes. Im 3 Kapitel geht es insbesondere um das theoretische Wissen welches benötigt wird um das scannen von Netzwerken zu verstehen. Bevor die eigentliche Evaluation der ausgewählten Netzwerkscanner im MWN ausgeführt wird der Versuchsaufbau dafür in 4 beschrieben.

Die gesamte praktische Leistung der Arbeit findet in Kapitel 5 statt, in der die definierten Kriterien anhand der Netzwerkscanner miteinander evaluiert werden. Gegen Ende der Arbeit werden in Kapitel 6 die aus dem Versuch gewonnen Messdaten dazu verwendet um die Praktikabilität der Scanner am MWN zu untersuchen. Abgeschlossenen wird in Kapitel 7, mit einer Zusammenfassung und einen Ausblick auf das Thema der Arbeit.

# 2 Grundlagen

Im dritten Kapitel dieser Arbeit werden die Grundlagen gelegt, die benötigt werden, um die eigentlichen Versuche im MWN verstehen zu können. Dabei werden die wichtigsten Protokolle, die für den Austausch von Informationen im Internet verwendet werden in ihren jeweiligen Unterkapiteln kurz vorgestellt.

Das Wissen darüber wird im Hauptteil der Arbeit benötigt, um die Ergebnisse der verschiedenen Netzwerkscanner zu verstehen. Dabei werden in diesem Kapitel nur Grundlagen erläutert, die nach Meinung des Autors relevant sind. Es ist also beabsichtigt, das manche Spezifikationen in einem genannten Protokoll nicht auftauchen, da sie für diese Arbeit keine wichtigen Informationen bereitstellen.

## 2.1 IPv6

Aufgrund der Probleme von IPv4 (hauptsächlich kleiner Adressenraum) wurde IPv6 als Nachfolger Protokoll entwickelt. Die Internet Engineering Task Force (IETF) definierte so einen neuen Standard, welcher die Schwächen seines Vorgängers beseitigen sollte. Spezifiziert wurde IPv6 in [DH17]. Das nachfolgende Unterkapitel wird das benötigte Wissen für diese Bachelorarbeit in Bezug auf IPv6 aufzeigen. Bevor das Protokoll im Detail näher erläutert wird, liefert dieser Abschnitt als näherer Einstieg genauere Unterschiede zwischen den beiden Protokollen.

Die Autoren von [DH17] listen in erster Linie fünf Änderungen in IPv6 zu seinem Vorgänger auf.

**Erweiterung des Adressraums** Der Hauptgrund für die Entwicklung von IPv6 bestand in der Erweiterung des Adressenraums. Im Protokoll IPv6 wird so der Adressraum von 32 Bits in IPv4 auf 128 Bits erhöht. In IPv6 wurde nicht nur der Adressbereich vergrößert, sondern auch an der Vergabe und Identifikation der Adressen zu einem Knoten (Host, Gerät) hat sich viel getan. Genauere Informationen zu den eben beschriebenen Erneuerungen finden sich in 2.1.3.

**Vereinfachung des Header-Formates** Der Header von IPv6 ist im Vergleich zu seinem Vorgänger IPv4 vereinfacht worden. Einige Felder wurden so weggelassen oder sind optional. Der IPv6 Header hat eine feste Länge von 40 Bytes. Es sind zwei mal 16 Bytes für die Quell und Zieladresse vorgesehen und nur 8 Bytes für die allgemeinen Header Daten.

Der IPv4 Header hat dazu im Vergleich mindestens 20 Byte an Header Informationen. Die Abbildung **??** zeigt die beiden Header im Detail aufgelistet. Durch das Weglassen des optionalen Teils im Header in IPv6 sind IPv6 Pakete besser strukturiert als IPv4 Pakete. Die optionalen Informationen die für ein IPv6 Paket zum Versenden benötigt

werden, befinden sich in sogenannten Erweiterungsheadern (eng. : Extension Headers). Dadurch das der Header von IPv6 immer eine feste Länge hat, ist es für Router einfacher, die benötigten Informationen auslesen zu können.

**Bessere Unterstützung für Erweiterungen und Optionen** Durch die Erweiterung der IPv6 Packten mit Extension Headern, ergeben sich im Vergleich zu IPv4 einige Vorteile. Es gibt somit keine Größenbeschränkung für Optionen im Vergleich zu IPv4, da zusätzliche Optionen einfach eingefügt werden, in dem ein Extension Header zu einem IPv6 Paket hinzugefügt wird. Jedoch sei zu beachten, dass je mehr Platz die Extension Header eines IPv6 Pakets wegnehmen, desto weniger Raum bleibt für den eigentlichen Payload (Nutzdaten). Außerdem können mit einer festen Header Größe auch Pakete schneller weitergeleitet werden.

**Flow-Label-Feld** Das Flow Label Feld (Fließetikett) wird spezifiziert in [ARCJ11]. Ein Flow ist eine Sequenz von Paketen, welche aus einer bestimmten Quelle an eine bestimmte Adresse gesendet werden. Eine Quelle kann dabei das 20 Bit Flow Label Feld in einem IPv6 Adresse benutzten, muss dies aber nicht, falls das Flow Label Feld aber benutzt wird müssen alle Pakete, die zu einem gleichen Flow gehören mit derselben Quelladresse und Zieladresse versehen werden. Außerdem gibt es einige Router und Hosts, die das Konzept des Flow nicht unterstützen, diese setzen das Flow Label Feld auf 0. Bei dieser Spezifikation muss sich noch zeigen, wie genau dieses Konzept am besten eingesetzt werden kann.

**Authentifizierung und Datenschutz** Mithilfe der Einführung von Erweiterungen ist IPv6 flexibler, was die Sicherheit betrifft. Zusätzliche Sicherheitsmechanismen wie die der Authentifizierung, Datenintegrität oder Vertraulichkeit der Daten können als Extension Header dem IPv6 Paket hinzugefügt werden. Denkbar ist es auch, neue Sicherheitsmechanismen als Extension Header hinzuzufügen, darin liegt der große Vorteil der Extension Header in dem Empfänger und Sender flexibel neue Erweiterungen hinzufügen können.

### 2.1.1 Darstellung

Eine IPv6 Adresse setzt sich aus 128 Bits zusammen, wobei die Adresse in 16 Bit Blöcken unterteilt wird. Jeder Block wird dabei von vierstelligen hexadezimal zahlen dargestellt.

Eine Adresse könnte zum Beispiel so aussehen:

**2001:4ca0:0:110::81bb:bf17**

Als Binäre Zahl ausgedrückt, würde die Adresse so aussehen:

0010000000000001 0100110010100000 0000000000000000 0000000100010000
0000000000000000 0000000000000000 1000000110111011 1011111100010111

Korrekt vollständig ausgeschrieben wäre die Adresse zwar 2001:4ca0:0:110:0000:0000:81bb:bf17, aber aufgrund von praktikablen Vorzügen hat man sich auf Abkürzen geneigt, wenn es darum geht IPv6 Adressen aufzulisten. Es gibt auch einige weitere Regeln zur Darstellung von IPv6 Adressen, welche in dem folgen Abschnitt kurz vorgestellt werden.

**Regeln zur Darstellung von IPv6 Adressen nach [KK10]:**

1. **Führende Nullstellen in einem 16-Bit Feld** Die führenden Nullen müssen durch :: ersetzt werden 20001:0db8::0001 ist nicht gültig richtig ist 2001:db8::1
2. **So viel wie möglich Kürzen** Es muss so oft wie möglich mit :: gekürzt werden. Zum Beispiel muss 2001:db8:0:0:0:0:2:1 zu 2001:db8::2:1 gekürzt werden
3. **Einen einzelnes 16 Bit 0 Feld nicht kürzen** Als Beispiel 2001:db8:0:1:1:1:1:1 ist richtig aber 2001:db8::1:1:1:1:1 nicht da ansonsten Informationen verloren gehen.
4. **Wahl bei der Platzierung von ::** Wenn es bei der Platzierung von :: Wahlmöglichkeiten gibt muss die längste Aneinanderreihungen von 16-Bit 0 Fields gekürzt werden
5. **Kleinbuchstaben** a, b, c , d ,e ,f müssen zur Eindeutigkeit als Kleinbuchstabe dargestellt werden

### 2.1.2 Adressaufbau

Eine 128 Bit IPv6 Adresse setzt sich aus mehreren Bereichen zusammen. Zu erst wird eine 128 Bit IPv6 Adresse in zwei Bereiche eingeteilt. Die ersten 64 Bits beziehen sich auf die Netzadresse und die letzten 64 Bits werden benutzt, um das Gerät eindeutig bestimmen zu können.Die 16 Bits der 64 Bit Netzadresse werden fürs Subnetting benutzt.

Als Beispiel betrachten wir die IPv6 Adresse:

**2001:4ca0:0:110::81bb:bf17 /64**

Zunächst teilen wir die Adresse in zwei Teile auf:

**2001:4ca0:0:110** und **0000:0000:81bb:bf17**

**2001:4ca0:0:110** ist also der Netzanteil und **0000:0000:81bbb:bf17** der Hostanteil. Man spricht aber nicht von Host und Netzanteil, sondern in IPv6 von *Network Präfix* und dem *Interface Identifier*. Dabei bezeichnet das Network Präfix das Netz, und der Interface Identifier den Host in diesem Netz.

Legen wir noch einmal einen genaueren Blick auf das Network Präfix:

**2001:4ca0:0:110**

An dem ersten Block erkennt man das es sich bei diesem Beispiel, um eine globale Adresse handelt, daher die Adresse ist aus dem gesamten Internet abrufbar. Bei einer globalen Adresse muss der erste Block mit 2001 beginnen. Insgesamt werden die ersten drei Blöcke, oder 48 Bits, benutzt um das Globale Routing Präfix festzulegen.

Ein Provider bekommt so einen fest geschriebenen Block erteilt. Für die Adressenvergabe zuständig ist in Europa Réseaux IP Européens (RIPE) welche dem genannten Beispiel einen fest verschriebenen Block festgelegt haben 2001:4ca0:0.

Der Letzte Block des Netzbereiches wird benutzt, um das Netz weiter verteilen zu können. Wie genau das Netz verteilt wird, kann dabei derjenige dem das Netz gehört frei entscheiden. Festgelegt ist nur, dass die letzten 16 Bits des Routing Präfixe benutzt werden für Subnetting.

### 2.1.3 Adressmodus und Gültigkeitsbereiche

Eine große Erneuerung in IPv6 im Vergleich zu IPv4 ist die Einführung neuer Adressmodi, während IPv4 nur zwischen privaten und öffentlichen Adressen unterscheiden konnte, können IPv6 Adressen mehrere Adressen besitzen, die einem Host gehören. Die verschiedenen Gültigkeitsbereiche können nach den verschiedenen Adressmodi vorgestellt in [DH06] eingeteilt werden.

### Unicast

Der Unicast- ist eine eindeutige Kennung für eine einzelne Schnittstelle. Wird ein Paket an eine Unicast- Adresse versendet wird, dieses Paket genau an die festgelegte Schnittstelle gesendet. Es gibt drei verschiedene Typen an Unicast-Adressen, wobei einer als veraltet gilt und nicht eingesetzt wird. Die drei verschieden Typen sind globale Unicast- Adressen, link lokale Unicast-Adressen und Site lokale Unicast-Adressen. Da Site lokale Unicast-Adressen veraltet sind, wird in dieser Arbeit nicht näher auf sie eingegangen.

**Interface Identifiers (Schnitstellen-ID):** Sowohl globale Unicast-Adressen als auch link lokale Unicast-Adressen verwenden Interface Identifiers (IID). Die IID werden benutzt, um einen Host in einem Netz mit seinem jeweiligen Subnetz fest identifizieren zu können.

**Global Unicast:** Als Beispiel für eine gültige globale Unicast-Adresse betrachten wir die Adresse in der Tabelle 2.1. Diese Adresse lässt sich in drei Teile unterteilen:

1. **Standortpräfix:** *2001:4ca0:0* ist der Standortpräfix der Adresse, diese beschreibt die öffentliche Topologie, die Ihrem Standort normalerweise von einem ISP oder einer Regional Internet Registry (RIR) zugewiesen wird. Es hat sich, als gängige Konvention herauskristallisiert, dass globale IPv6 Adressen immer mit dem Präfix 2001 beginnen.

2. **Subnetz:** 110 ist die Subnetz Adresse diese kann der Provider für sein Netz selbst frei einteilen. Dabei sind 16 Bit für das Subnetz vorgesehen. Insgesamt macht der Netzanteil einer globalen IPv6 Adresse somit 64 Bit aus.

3. **IID:** *0000:0000:81bb:fb17* die Restlichen 64 Bits werden dem IID zugeordnet.

Tabelle 2.1: Einteilung einer Globalen IPv6 Adresse

| **Standortpräfix** | **Subnetz** | **IID** |
|---|---|---|
| 2001:4ca0:0 | 110 | 0000:0000:81bb:fb1 |

**Link Lokale:**
Link Lokale Adressen werden benutzt für ein einzelnen Knoten damit dieser sich bei ersten Gebrauch selbst eine eigene Adresse Bilden kann und somit mit seinen angeschlossenen Netz und insbesondere mit seinem Router kommunizieren kann.

Als Beispiel für eine gültige lokale Unicast-Adresse nehmen wir die Adresse:

**fe80::250:56ff:fe8f:1d21**

Alle Adressen von **fe80::** bis hin zu **febf::** sind als Link Lokale Adressen identifiziert und werden nicht vom Router weitergeleitet. Die Letzten 64 Bits sind IID.

#### Anycast

Der Anycast wird mehreren Schnittstellen zugewiesen (normalerweise auf mehreren Knoten). Eine an eine Anycast-Adresse gesendetes Paket wird nur an eine dieser Schnittstellen gesendet. Normalerweise wird die Nachricht also an die nächste Verfügbare Adresse gesendet. Dabei wird die nächste Adresse im Routing Protokoll definiert.

Anycast ist gewährt die Verbindung zwischen einem einzelnen Sender und dem nächstgelegenen Empfänger in einer Gruppe. Anycast wurde entwickelt damit ein Host Router-Tabellen für eine Gruppe von Hosts effizient aktualisieren kann.

IPv6 findet dabei heraus, welches Gateway am nächsten ist und schickt die Pakete an diesen Host wie bei der Kommunikation mit Unicast. Dieser Host kann wiederum eine Anycast-Kommunikation mit einem anderen Host in dieser Gruppe starten. Die Kommunikation findet so lange statt, bis alle Routing-Tabellen aktualisiert sind.

#### Multicast

Die Multicast-Adresse ist ein Bezeichner für eine Reihe von Schnittstellen (die normalerweise zu verschiedenen Knoten gehören. Ein Paket an eine Multicast-Adresse wird an alle Schnittstellen gesendet die durch diese Adresse identifiziert werden. Zu seinem Vorgängermodell hat IPv6 keine Broadcast Adresse. Die Funktion des Broadcasts wird durch den Multicast abgelöst.

### 2.1.4 Kommunikation

In IPv6 gibt es keinen Broadcast wie ein IPv4. Im Gegensatz zu IPv4 kann ein IPv6 Host sich seine Adresse auch selbst generieren und muss diese nicht von einem DHCP Server bekommen oder manuell konfiguriert werden. Im folgenden Abschnitt werden typische Schritte die ein neuer Host in einem IPv6 Netzwerk durchläuft geschildert.

1. Wenn ein IPv6 Host ein Netzwerk beitritt, wird er als erstes versuchen einen Router in seiner lokalen Umgebung zu finden.
2. Um einen Router zu findet sendet, dieser ein Router Solicitation (RS) Paket.
3. Das RS Paket wird an die Multicast-Adresse FF02::2 gesendet (an dieser Adresse befindet sich alle Router eines jeweiligen Netzes).
4. Im Header des Router Advertisment Paket (RA) Pakets befindet sich die Quell Adresse und als Zielnetz die Multicast-Adresse FF02::2.
5. Als Antwort sendet der Router dem Host ein RA Paket.
6. Je nach Einstellung sendet der Router auch periodisch ein RA Paket an alle Knoten mithilfe der Multicast-Adresse FF02::1

7. Mit der Information aus dem RA hat der Host alle Daten um eine IPv6 Verbindung aufzubauen

### 2.1.5 ICMPv6

Gerade für das Untersuchen eines IPv6 Netzwerkes ist das Protokoll ICMPv6 (Internet Control Messsage Protocol ) von großer Bedeutung. Vereinfacht gesagt dient ICMPv6 ähnlich wie ICMP für IPv4 in Netzwerken zum Austausch von Fehlern und Informations- Meldungen. Dabei kann ICMPv6 als Hilfsprotokoll für IPv6 betrachtet werden. So können ICMPv6 Nachrichten über IPv6 versendet werden. Die Hauptspezifikation dieses Protokolls wird im RFC [GC06]vorgenommen.

**Nachrichten Format** Jeder ICMPv6 Nachricht geht ein IPv6 Header voraus mit null oder weiteren IPv6 Erweiterungsheadern. Der ICMPv6 Kopf wird von einem Next Kopf mit dem wert 58 identifiziert.

**ICMPv6 Kopf:** In Tabelle 2.2 ist der Kopf einer ICMPv6 Nachricht zu sehen und im folgenden werden kurz die einzelnen Attribute beschrieben.

**Type** Gibt den Typ der Nachricht an. Dieser Wert bestimmt das Format der restlichen Daten

**Code** Ist abhängig vom Nachrichtentyp. Es wird verwendet, um eine zusätzliche Ebene der Nachrichtengranularität zu schaffen.

**Prüfsumme** Wird verwendet um Datenverluste in einer ICMPv6 Nachricht und in Teilen des IPv6 Kopfes zu erkennen.

**ICMPv6 Nachricht** [GC06] Klassifiziert ICMPv6 Nachrichten in zwei Gruppen, zu einem in Fehlermeldungen und in informativen Nachrichten. Dabei gehen Fehlermeldungen von 0 bis 127 und informative Meldungen von 128 bis 255. Tabelle 2.3 zeigt einige der wichtigen Meldungen.

Tabelle 2.2: ICMPv6 Kopf

| 0 -7 Bits | 8-15 Bits | 16-23 Bits | 24-31 Bits |
|---|---|---|---|
| Type | Code | Prüfsumme | |
| ICMPv6 Nachricht | | | |

Tabelle 2.3: Wichtige ICMPv6 Nachrichten

| **Error Messages** | **Informational Messages** |
|---|---|
| Destination Unreachable | Echo Request |
| Packet Too Big | Echo Reply |
| Time Exceeded | |
| Parameter Problem | |

### 2.1.6 Adressen Vergabe

In IPv6 gibt es grundsätzlich zwei verschiedene Arten Adressen zu vergeben, die miteinander kombiniert werden können. Zum einem gibt es die Möglichkeit der Vergabe per Stateless Adress Autoconfiguration (SLAAC) oder die per Dynamic Host Configuration Protocol for IPv6 (DHCPv6).

Dabei handelt es sich bei SLAAC bei einem stateless (zustandslos) Adressen Konfiguration und bei DHCPv6 bei einer stateful (Zustandshaltend) Vergabe von Adressen. In den meisten gängigen Systemen wird eine Kombination von SLAAC und DHCPv6 zu Adressen Vergabe angewandt. In dem folgenden Abschnitt werden alle möglichen Varianten der Adressen Vergabe genau erklärt.SLAAC wird in [NJT07] definiert und DHCPV6 in [PVL+03]

**SLAAC** Die Grundidee von SLAAC ist es, dass jeder Client in einem IPv6 Netzwerk die Möglichkeit hat seine Adresse selbst zu konfigurieren. Dies geschieht in dem er am Anfang ein ICMPv6 Router Solicitation Messsage an eine Multicast-Adresse verschickt für die Parameter, die er benötigt, um sich seine Adresse zu konfigurieren.

Der lokale Router erwidert so eine Anforderung indem er, ein Router advertisment Message verschickt, welches die benötigten Parameter enthält. In einer Router advertisment Message befinden sich alle benötigten Konfigurationsinformationen für den Client. Die wichtigste Information für den Client aus dieser Nachricht ist die des IPv6 Präfixes, welche der Client hernehmen sollte, um sich seine lokale Adresse zu bilden. Mithilfe des Präfixes generiert der Client seinen lokalen Interface Identifier (IID)

**DHCPv6** DHCPv6 liefert die Möglichkeit, Adressen mit festgeschriebenen Richtlinien fest in einem Netz zu vergeben. Im Gegensatz zu SLAAC, welche Adressen nur mithilfe von angeschlossenen Routern vergeben konnte benötigt DHCPv6 einen festen Server, der die Adressen in einem Netz vergibt.

Ähnlich wie in IPv4 vergibt der Server Adressen fest aufgrund einer Konfigurationsdatei, welches die Richtlinien der Adressenvergabe (z. B mögliche Reichweite in der einen Adresse liegen darf) fest vergibt.

**Manuell Konfigurierte Adressen** Manuell konfigurierte Adressen tauchen hauptsächlich bei Routern (haben keine Funktion, um automatisch Adressen sich selbst zu konfigurieren) und Servern (Adresse sollte nicht von der unterliegenden link-layer Adresse abhängig sein) auf.

# 3 Netzerkscannen

## 3.1 IPv6 Netzwerkscannen

Im vorherigen Kapitel wurden alle wichtigen Grundkenntnisse aus dem IPv6 Protokoll gelegt. Dieses Kapitel beschäftigt sich mit dem benötigten Wissen für das konkrete Scannen von Adressen in einem IPv6 Netz. Das Hauptwissen aus diesem Kapitel kommt dabei aus [GC16].

### 3.1.1 Lokale Netzwerke

Das Scannen von lokalen Netzwerken ist deutlich einfacher als das von externen. Voraussetzung ist das der Scanner von einem Host ausgeführt wird, welcher sich im gleichen Lokalen Netzwerk befindet. Mithilfe des Sendens eines Prüfpaketes an alle Link-Lokal Knoten aller Multicast Adressen wird der lokale Scan ausgeführt. Sodass alle angeschlossenen Geräte des Netzes auf diese Anfrage antworten. Ein kleines Problem bei dieser Herangehensweise sind Windows Systeme die nicht auf ICMPv6 Echo Anfragen an Multicast Adressen reagieren.

Dies liegt an der Spezifikation in [GC06] dort steht nämlich

*An Echo Reply SHOULD be sent in response to an Echo Request message sent to an IPv6 multicast or ancast address.*

Es muss also von einem Host nicht auf ein Echo Anfrage geantwortet werden, die auf eine Multicast Adresse adressiert ist. Um auch die Adressen von Windows Systemen erhalten zu können, senden Scanner weitere Prüfungspakete, um von allen lokalen Knoten eine Antwort bekommen zu können. Es werden z.B ICMPv6 Nachrichten mit nicht erkannten Typen verschickt worauf dahin Hosts als Nachricht ein Parameter Problem verschicken.

### 3.1.2 Externe Netzwerke

In IPv4 wurden Remote Netzwerke noch in absehbarer Zeit per Brute Force Attacken gescannt. Aber in IPv6 ist es nicht möglich in endlicher Zeit alle $2^{64}$ Adressen von IPv6 zu scannen. Es müssen also neue Strategien her, um weiterhin IPv6 Netze scannen zu können. Eine mögliche Strategie, die in [GC16] erläutert wird, ist es allgemein sich zu überlegen wie man den möglichen Adressraum verkleinern kann, um nicht /64 Netze scannen zu müssen.

**Beispiele um den durchsuchten Adressraum zu verkleinern nach [GC16]:**

1. **Eingebettete IPv4 Adressen:** Eine Form nach der gesucht werden kann, ist die der Einbettung der IPv4 Adresse. So wird die bestehende IPv4 Adresse die, als Dezimalzahl beziffert ist in eine Hexadezimal Ziffer umgewandelt. Üblich werden dann die letzten 32 Bits der IPv6 Adresse mit der umgewandelten Zahl belegt.

2. **Dienst Port Adressen:** Der Zahl des Ports kann benutzt werden um am Ende eine IPv6 Adresse zu generieren z.B 2001:4ca0::80

3. **Wortreiche Adressen:** Durch die Hexadezimalzahlen können Wörter in IPv6 Adressen erstellt werden wie z.B 2001:4ca0::dead:beef

4. **SLAAC Identifier:** Adressen werden durch ein automatisches Verfahren erstellt und können so falls die Netzwerkkarten bekannt sind berechnet werden.

5. **Low Byte:** Als Low Byte werden die letzten 24 Bytes einer IPv6 Adresse bezeichnet in dieser Region befinden sich oftmals aus Bequemlichkeit einige Server.

## 3.2 Netzwerk testen

Ein Hauptziel dieser Arbeit ist es Netzwerkscanner im IPv6 Netzwerk des MWN vergleichen zu können. Bevor der eigentliche Vergleich stattfindet, kann, muss aber erst geklärt werden welche Funktionalitäten, ein Netzwerk Scanner bereitstellen muss. Die Benutzung von Netzwerkscanner unterliegt allgemein zwei Zielgruppen.

Zum einen gibt es den potenziellen Angreifer, der durch gängige Netzwerkscanner potenzielle Schwachpunkte seines Zielnetzes oder Hosts herausfinden möchte. Im Gegensatz dazu es die Rolle des Netzwerkadministrators, der zum Ziel hat sein Netzwerk mit den angeschlossenen Geräten möglichst sicher halten zu können.

Beide Zielgruppen haben unterschiedliche Absichten wofür sie die Netzwerkscanner benutzen möchten beide bedienen sich derselben Methoden und Techniken, um ihre Ziele realisieren zu können. Im folgenden Abschnitt werden alle Grundlagen gelegt, die benötigt werden, um zwei große Abschnitte eines Penetrationstests oder Sicherheitstest vollziehen zu können.

Der erste wichtige Abschnitt ist der Host Erkennung. Die Host Erkennung, dient dazu möglichst ein breites Spektrum an Hosts eins Netzwerks finden zu können. Der nächste Abschnitt der des Port Scannens dienst dazu mögliche Schwachstellen einer Maschine oder sogar eines ganzen Netzwerks finden zu können. Aus Zeitgründen wird so erst eine Liste mit potenziellen Hosts eines Netzwerkes erstellt, mit der Liste der Hosts, welche eine IP Adresse ist, werden danach die Ports gescannt. Dies geschieht in dieser Reihenfolge, um Zeit sparen zu können, und kann deshalb auch getrennt voneinander betrachtet werden.

### 3.2.1 Host Discovery

Im ersten Abschnitt überprüft ein Netzwerkscanner anhand eines Netzwerkes oder einem gegeben Adressen Bereich welche Adressen des zu untersuchenden Bereiches wirklich aktiv sind. Wenn sich jeder Host an [Bra89] halten würde wäre es eine Leichtigkeit für Netzwerkscanner zu überprüfen ob sich hinter einer IP Adresse wirklich in Host befindet. Dort steht nämlich:

*Every host MUST implement an ICMP Echo server function that receives Echo Requests and sends corresponding Echo Replies*

Und zwar steht im RFC 1122 das jeder Host auf einen Echo Request, also einen einfachen Ping Antworten muss. Dies ist aber auch eine Sicherheitslücke weshalb viele Administratoren ihr Netzwerk so eingerichtet haben, das Hosts nicht auf einen Ping antworten. Insbesondere die Windows Firewall blockt standardmäßig solche Anfragen, das heißt mit einem einfachen Ping können so viele Hosts in einem Netzwerk nicht bestimmt werden.

Da sich diese Bachelorarbeit aber mit dem Scannen von IPv6 Adressen beschäftigt bleibt die Frage offen ob es auch gängig ist das Hosts im IPv6 Adressraum nicht auf Echo Requests antworten. Das RFC [DM07] liefert hier eine Antwort in der sie das Argument des großen Adressraumes liefert welche IPv6 Adressen davor schützt zu viele Anfragen zu bekommen. Außerdem ist ICMPv6 in IPv6 wichtiger als in IPv4 und in diesem RFC wird dazu geraten das Firewalls keine Echo Request Anfragen filtern sollten.

Für IPv4 Adressen verwenden Netzwerkscanner aber deshalb nicht nur ein einfachen Ping zum auffinden von Hosts, sondern eine Vielzahl an unterschiedlichem Anfragen. Zum Beispiel verwendet Nmap ein ICMP Echo Request mit einem TCP SYN auf Port 443 und einem TCP ACK auf Port 80 und ein ICMP timestamp Request Das ICMP timestamp Request ist aber nur für IPv4 gängig. Daneben schickt ein weiter Netzwerkscanner Scan6 neben dem einfachen Ping ein IPv6 Packet mit unerkannten Optionen damit insbesondere Windows Systeme, die nicht auf Echo Request antworten trotzdem eine Antwort schicken.

Im folgenden Abschnitt werden die unterschiedlichen Host Discovery Techniken näher erläutert (Definitionen entstammen dabei aus [Lyo09]):

- **TCP SYN PING:** : Bei einem TCP SYN Ping baut ein Host zunächst eine normale TCP Verbindung mit dem Ziel auf. Als Port kann ein beliebiger Port gewählt werden, meistens versucht man aber einen Port zu wählen der Wahrscheinlich häufig im Zielnetz auftrifft. Zu nächst schickt der Client an dem Ziel host ein TCP packt indem das SYN Flag gesetzt ist. Wenn der Port offen ist schickt der Ziel host im zweiten Schritt als Antwort ein SYN ACK TCP Paket Normalerweise würde jetzt wieder der Host ein ACK zurückschicken und eine TCP Verbindung wäre aufgebaut. Da das Ziel host aber nicht die Absicht hat eine Verbindung aufzubauen, sondern nur überprüfen will ob sich hinter der IP Adresse in Host befindet wird ein RST zurückschickt, welche dem Ziel host mitteilt das es den Verbindungsversuch vergessen soll. Nichts zu schicken wäre schlecht da der Ziel host an. Wenn der Zielport geschlossen ist wird ein RST zurück geschickt. Zusammengefasst wenn der Scanner beim zweiten schritt ein SYN/ACK bekommt ist der Port offen, wenn ein RST zurück kommt ist der Port geschlossen.

- **TCP ACK PING:** : Bei einem Ack Ping wird nicht wie bei einem SYN Ping am Anfang ein SYN Flag an das Ziel geschickt, sondern ein ACK. Es wird also vorgetäuscht das schon eine Verbindung besteht, darauf hin schickt das Ziel in der Regel in RST welches seine Existenz verrät. Der Grund für diesen Scan sind Firewalls die häufig ein SYN auf Ports blockieren, die nicht öffentlich zugänglich sein sollen, hier kann dieser Scan den Port doch ansprechen und feststellen ob ein Host im Netz ist.

- **UDP PING:** Bei einem UDP Ping wird an einen ausgewählten Port ein leeres UDP Packet geschickt, wenn der Port geschlossen ist sollte ein ICMP port unreachable Packet zurück kommen. Damit weiß der Netzwerkscanner das die Maschine im Netz ist. Falls ein offener Port erreicht wird kann es sein das der Ziel Host keine Antwort zurück schickt da das Packet leer war. Deshalb bietet es sich bei UDP an keine Standartports zu benutzen. Ein Vorteil dieses Ping ist es Firewalls zu entgehen die nur TCP Pakete filtern.

- **ICMP Ping Types:** Die meisten Netzwerkscanner können auch Standard Pings senden, wie ein ICMP Type 8 (Echo Request) an ein Ziel und erwarten dann als Antwort ein Echo Replay. Analog dazu für ICMPv6.

- **IP Protocol Ping:** Diese Methode der Host-Erkennung sucht nach Antworten, die entweder dasselbe Protokoll wie der Test haben, oder Meldungen, dass das ICMP-Protokoll nicht erreichbar ist, was bedeutet, dass das gegebene Protokoll vom Zielhost nicht unterstützt wird. Beide Antworten bedeuten, dass der Zielhost am Leben ist.

- **Kombination aus verschiedenen Ping Arten:** Mit einem einzelnen Scan kann man auch eine Kombination aus mehreren Ping Scans durchführen, gerade wenn man weiß, dass gewisse Ports in einem Netzwerk vorhanden sind und so eine Antwort liefern kann man TCP UDP und einfache ICMP Pings miteinander kombinieren.

### 3.2.2 Port Scannen

Nachdem bestimmt wurde ob eine Adresse in einem Netzwerk vergeben ist und sich ein Host hinter diese Adresse befindet beginnt die eigentliche Überprüfung möglicher Sicherheitsschwachstellen indem ein Netzwerkscanner die Ports seines Zielhostes durchsucht. Durch Ports wie z.B Port 80 welcher benutzt wird um http Anfragen zu verschicken kann ein Computer Informationen abrufen oder auch Informationen bekommen. Dadurch sind Ports eine Art Tür zu einem Zielhost deshalb bedarf es zur Sicherheit die Überwachung alles Ports. Es kann oft vorkommen das manche Programme standardmäßig Ports öffnen von denen der Betreiber gar nicht genau weiß. Diesen offenen Port können dann Angreifer ausnützen um Zugang zum System zu bekommen. Dabei kann ein Host insgesamt 65535 verschiedene Ports in UDP und TCP besitzen. Möchte man nun alle Ports eines Zielnetzwerkes untersuchen in UDP und TCP müsste man 131070 Anfragen an einen Host jeweils mit TCP oder UDP schicken. Zwar hätte man so eine Vollständige Untersuchung eines Netzwerkes aber aus Zeitgründen ist es oft besser nur die 1000 häufigsten oder sogar nur 100 häufigsten benutzen Ports zu benutzen. Ein Scanner wie Nmap scannt so falls nicht weiter vorgegeben Standardmäßig die 1000 häufigsten Ports eines Hosts. Die Autoren der Arbeit von [BLRS19] haben Port Scans anhand vier unterschiedlicher Typen eingeteilt, wobei für diese Arbeit drei dieser Typen verwendet werden, um Port Scans genauer definieren zu können. Die Port Zustände wurden explizit für Nmap definiert, d.h nicht alle Netzwerkscanner bedienen sich dieser Klassifikation. Für manche gibt es auch nur einen Zustand (offen). Die Definitionen entstammen aus [Lyo09]

**Port-Typen und Zustände:**

**vertikal** Der vertikale Port Scan ist ein Scan, welcher mehrere Ports eines einzelnen Hosts untersucht.

**horizontal** Ein horizontale Port Scan ist ein Scan, welcher an mehreren Hosts nur einen einzelnen Port untersucht.

**block** Ein blockscan ist eine Kombination aus einem vertikalen und horizontalen Port Scan. Das bedeutet das mehrere Ports an mehreren Hosts untersucht werden.

**offen** Ein Programm ist bereit TCP/UDP Pakete auf diesem Port anzunehmen.

**geschlossen** Ein geschlossener Port ist erreichbar, aber es gibt kein Programm das ihn abhört.

**gefiltert** Es kann nicht festgestellt werden ob der Port offen ist weil in Filter verhindert das Pakete den Port erreichen.

**ungefiltert** Bedeutet das der Port zugänglich ist aber Nmap nicht feststellen kann ob er offen oder geschlossen ist.

**offen|gefiltert** Nmap klassifiziert einen Port in diesen Zustand, wenn es nicht feststellen kann, ob der Port offen oder gefiltert ist.

**geschlossen|gefiltert** Dieser Zustand wird benutzt, wenn Nmap nicht feststellen kann, ob ein Port geschlossen ist oder gefiltert wird.

### Verschiedene Port Scan Arten:

Es gibt eine Reihe an unterschiedlichen Port Scan Methoden im folgenden werden die einzelnen beschrieben. Diese entstammen auch aus dem Buch [Lyo09]

**TCP SYN:** Bei einem SYN Scan versucht der Scanner eine normale TCP Verbindung mit dem Ziel aufzubauen, wenn das Ziel ein SYN/Ack zurückschickt ist der Port offen während bei einem RST der Port, als geschlossen gilt. Es wird aber keine vollständige TCP Verbindung aufgebaut, wenn der Port offen ist, da der Scanner im dritten Schritt nicht wie üblich ein SYN Ack zurückschickt um eine TCP Verbindung aufzubauen, sondern ein RST welches die Verbindung zurücksetzt.

**TCP Connect scan** Im Gegensatz zum SYN Scan geht der Connect Scan eine echte TCP Verbindung mit dem Ziel ein.

**TCP Null, XMAS, FIN scan:** Allgemein nutzen diese Scans das TCP Protokoll aus, um zwischen offenen und geschlossenen Ports unterscheiden zu können. Dies passiert indem der TCP-Flag-Header unterschiedlich gesetzt wird. Bei einem Null TCP Scan sind alle Kopf Header im TCP Packet auf 0 gesetzt. Diese Pakete sind nicht erlaubt und bei offenen Ports werden sie ignoriert. Geschlossene Ports Antworten hingegen mit einem RST Packet. der FIN Scan setzt nur das TCP-FIN Bit. Der XMAS Scan setzt die FIN PSH und URG Flags.

**TCP ACK:** Der TCP ACK Scan unterscheidet sich in den anderen insoweit dass, er nicht versucht offene oder geschlossene Ports zu bestimmen, sondern dient dazu Firewall Regeln bestimmen zu können und welche Ports gefiltert werden. Beim Paket eines ACK Scan wird dabei nur das ACK Flag gesetzt. Beim Scannen eines ungefilterten Hosts werden dabei sowohl offene als auch geschlossene Ports ein RST Paket zurückgeben. Ein Scanner wie Nmap markiert diese dann als ungefiltert, dass bedeutet sie werden zwar vom Paket erreicht aber es kann nicht bestimmt werden, ob diese offen oder geschlossen sind. Ports die nicht antworten oder ICMPv6 Fehlermeldungen zurückgeben werden als gefiltert markiert.

**UDP Scan** UDP Scans können nur bestimmen ob ein Port geschlossen ist geschlossene Ports antworten mit einem ICMPv6 Paket. Offene Ports können nicht bestimmt werden da das Ziel nicht auf diese Anfragen angeht. Meistens haben die UDP Pakete auch keinen Inhalt und das Ziel verwirft somit dieses Paket.

# 4 Versuchsaufbau

Bevor es zur Evaluation der verschiedenen Netzwerkscanner kommt, werden in diesem Kapitel alle Rahmenbedingungen geklärt. Diese beinhalten das Versuchsnetz, die Versuchsmaschine und die ausgewählten IPv6 Scanner, die in dieser Arbeit betrachtet werden. Außerdem werden die zu untersuchenden Kriterien geklärt.

## 4.1 Rahmenbedingungen

Da diese Bachelorarbeit am Leibniz Rechenzentrum verfasst wurde, benutzt die Arbeit als zu untersuchendes Netz den IPv6 Adressbereich des MWN. Des Weitern wurde sich nicht nur aufgrund der Tatsache, dass diese Arbeit am LRZ verfasst wird, dafür entschieden das MWN als Zielnetz auszuwählen. Sondern ein so großes Netzwerk wie das des LRZ eignet sich gut um IPv6 Netzwerke und Probleme die sich zum Scannen des IPv6 Adressbereiches ergeben zu untersuchen. Der Rechner, aus dem die Netzwerk Scans und die weiteren Methoden getestet werden ist eine virtuelle Maschine am LRZ. Im Jahr 2005 hat das LRZ sich durch eine Mitgliedschaft bei RIPE einen globalen routebaren IPV6 Block gesichert. Konkret bedeutet das, dass alle globalen IPv6 Adressen die mit 2001:4ca0::/32 beginnen dem MWN zugeordnet werden können.

### 4.1.1 Zielnetz

Das Münchner Wissenschaftsnetz (MWN) verwaltet die Netzstruktur der Münchner Hochschulen. Außerdem werden viele weitere Einrichtungen vom MWN verwaltet. Genauere Auflistung über den gesamten Bereich der vom MWN verwaltet wird und weitere ausführliche Informationen liefert [HR16]. Da sich diese Arbeit hauptsächlich mit dem IPv6 Protokoll befasst, werden in diesem Unterkapitel nur alle relevanten Informationen, die die Arbeit mit IPv6 betreffen näher erläutert. Insgesamt gibt es im MWN über 900 Subnetze wobei die meisten sowohl IPv4 und IPv6 benutzen, so sind nur einige wenige Legacy Systeme nur über IPv4 erreichbar. Die IPv6 Adressen werden so für neue Hosts über SLAAC vergeben. Die Serverrechner erhalten zur besseren Wartung manuelle Adressen. Zum besseren Verständnis für den Umfang des Netzes sei auf 4.1 verwiesen.

### 4.1.2 Versuchsmaschine

Für die praktischen Versuche dieser Arbeit wurde eine Virtuelle Maschine (VM) die sich im MWN Adressraum befinden eingerichtet. Die globale IPv6 Adresse der VM ist 2001:4ca0:0:110::81bb:fb17. Die VM teilt sich die Hardware mit weiteren VM. Die Tabelle 4.1 liefert eine genaue Auflistung der Hardware der VM.

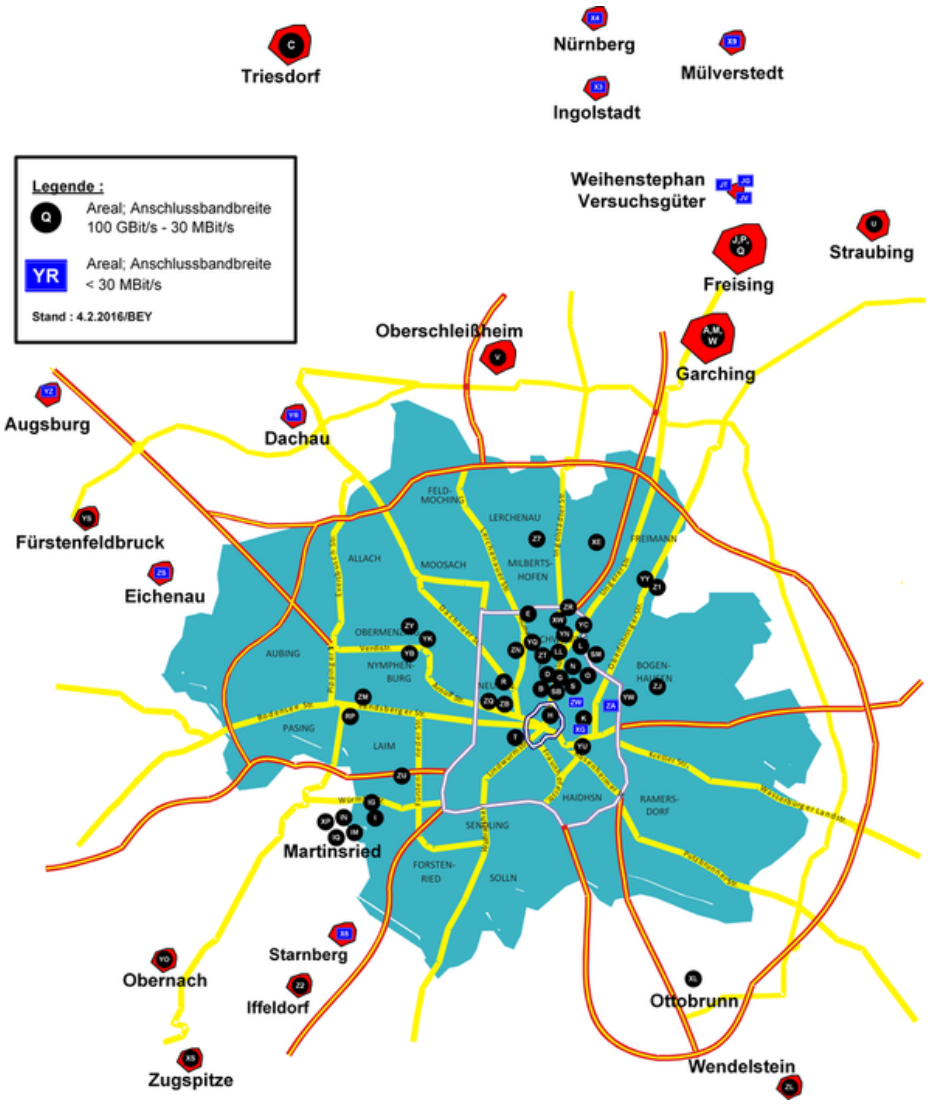



Abbildung 4.1: Die Karte des MWN

Tabelle 4.1: Die Hardware Daten der VM

| | |
|---|---|
| **CPU Model** | Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz |
| **CPU(s)** | 2 |
| **Thread per core** | 1 |
| **Core per socket** | 1 |
| **RAM** | 8 GB |
| **OS** | Ubuntu 16.04.6 LT |
| **System** | VMware Virtual Platform |
| **Network** | VMXNET3 Ethernet Controller |
| **size** | 10Gbit/s |
| **width** | 32 bits |
| **clock** | 33MHz |

## 4.2 IPv6 Scanner

Bevor es zur eigentlichen Evaluation der Scanner kommt, dient dieses Unterkapitel als grobe Übersicht über aktuelle Scanner. Außerdem wird in diesem Abschnitt die Entscheidung begründet weshalb die gewählten Netzwerkscanner ausgewählt worden sind, die mit praktischen Versuchen näher evaluiert wurden.

### 4.2.1 Überblick über aktuelle Scanner

#### Nmap

Nmap steht für Network Mapper und dürfte wohl der bekannteste Netzwerkscanner sein. Insbesondere wird Nmap dazu verwendet Ports eines Netzwerkes scannen zu können und damit verbunden mögliche Sicherheitsschwachstellen ausfindig machen zu können. Auch im LRZ ist Nmap im Einsatz um im IPv4 Adressraum mögliche Sicherheitsschwachstellen finden zu können.

Für den IPv4 Bereich liefert Nmap alle Optionen die ein Administrator oder potentieller Angreifer brauchen kann. Seid 2002 unterstützt Nmap auch IPv6. So funktionieren viele Scans für IPv4 in Nmap ähnlich wie in IPv6 nur das der Anwender in dem gewünschten Befehl -6 davor schreiben muss. So scannt zum Beispiel der Befehl Nmap -6 -sv 2001:4ca0::1 die Ports der IPv6 Adresse.

Auch einzelne Adressen können mithilfe von Nmap -6 -sP 2001:4ca0::1 gescannt werden. Mit Nmap lassen sich viele Funktionen ausführen (Host-Erkennung, Port scannen, Versionserkennung von Diensten, Betriebssystemerkennung). Außerdem ist es möglich eigene Skripte für Nmap zu schreiben. Mithilfe der Nmap Scripting Engine (NSE) ist es so möglich zusätzliche Funktionalitäten in Nmap einbauen zu können. Auch für IPv6 gibt es eigene Skripte.

#### Zmap / Zmapv6

Zmap wurde aus der Motivation heraus entwickelt, schnelle breite Internet Scans ausführen zu können. In dem Paper zu Zmap [ZD13] gehen die Autoren auf die Architektur von Zmap ein und vergleichen diese auch mit Nmap. So kann im IPv4 Adressbereich Zmap den ganzen Adressbereich in unter 45 Minuten scannen. Außerdem haben die Autoren in ihrer Arbeit Messdaten für Nmap und Zmap aufgenommen und konnten zeigen, das Zmap 1300-mal schneller ist als Nmap. Im Gegensatz zu Nmap kann Zmap aber nur einzelne Ports durchsuchen und unterstützt zurzeit TCP SYN ICMP ECHO und UDP Scans.

Nach Nmaps Beispiel ist es für Nutzer auch möglich eigene Module für Zmap zu schreiben. In der Arbeit von [GSGC16] haben die Autoren eine große Ansammlung an IPv6 Adressen erstellt die es anderen Forschern ermöglichen sollte weiter an IPv6 Adressen zu forschen.

Um überprüfen zu können, ob die aus unterschiedlichen Methoden gewonnen IPv6 Adressen auch wirklich existieren oder noch verfügbar sind haben die Autoren Zmap erweitert. So haben sie drei neue Module geschrieben mit denen es möglich ist IPv6 Adressen zu untersuchen. Da die Architektur von Zmap interessant ist, wird hier kurz auf sie eingegangen damit Leser der Arbeit verstehen können, wieso Zmap so schnell ist.

**Zusammenfassung der Architektur von Zmap aus [ZD13]:**

**Optimizied probing** Nmap passt seine Datenübertragung an um Ziele nicht zu überfordern, Zmap macht dies nicht. Bei Zmap wird auch der TCP/IP Stack übergangen und Ethernet Frames werden direkt verschickt. Zmap verschickt aus einer Liste von IP Adressen oder Bereichen (nur in IPv4) nicht nacheinander die Anfragen sondern die Reihenfolge wird zufällig in der Liste gemacht.

**No per connection state** Nmap speichert den Status von jeder Verbindung die es versucht aufzubauen. Im Gegensatz dazu speichert Zmap den Status nicht, Zmap ermittelt auch keine Verbindung timeouts.

**No retransmission** Nmap merkt sich wenn ein Ziel nicht antwortet und schickt nochmal Proben, Zmap hingegen schickt immer eine feste Anzahl an Proben Standardmäßig ist das nur eine.

#### Masscan

Der Netzwerkscanner Masscan vereint Nmap und Zmap indem es ähnlich wie Nmap alle Ports eines Hosts scannt, aber im Gegensatz zu Nmap die schneller scannen kann. Masscan bietet aber keinerlei Funktionen für IPv6.

#### Scan6

Scan6 ist ein IPv6 fähiger Netzwerkscanner und wurde von Fernando Gont entwickelt. Der Autor ist Beruflich ein Schachstellentester und hat viele weitere Werkzeuge geschrieben, diese sind alle zusammen gefasst als IPv6toolkit [si6]. Eins dieser Werkzeuge ist Scan6, welcher als IPv6 Netzwerkscanner benutzt werden kann. Insbesondere bedient dieser Scanner viele Möglichkeiten einen IPv6 Adressraum nach Mustern durchsuchen zu können.

#### Alive6

Marc Heuse ist wie Fernando Gond ein Sachwachstellentester und hat eine Hand von Werkzeugen entwickelt, um Penetrationstests gegen IPv6 Netzwerke ausführen zu können. Diesem enstammt auch der IPv6 Netzwerkscanner Alive6.

#### Chiron

Wurde von Antonios Atlasis entwickelt und ist auch wie Scan6 und Alive6 als Penetration Tool gedacht, mit dem der Angreifer insbesondere im IPv6 Bereich einige interessante Angriffe starten kann.

### 4.2.2 Auswahl geeigneter Scanner

Das Hauptkriterium der Scanner, welche für diese Arbeit von Bedeutung ist, ist die Unterstützung von IPv6, weswegen ein Scanner wie Masscan als erstes herausfällt. Daneben unterstützten die Netzwerkscanner Nmap, Zmap, Alive6, Scan6 und Chiron alle IPv6 und sind auch noch Open Source. Es wurde sich deshalb in der vorliegenden Arbeit dafür entschieden, alle vier dieser Netzwerkscanner zu evaluieren.

## 4.3 Kriterien für den Vergleich der Netzwerkscanner

Um die Netzwerk Scanner miteinander vergleichen zu können ist es wichtig, sich vorher darüber Gedanken zu machen, nach welchen Kriterien diese verglichen werden sollen. Deshalb definiert dieser letzte Bereich der Implementierung, die Kriterien fest, die für alle fünf Netzwerkscanner evaluiert werden sollen.

### 4.3.1 Host Erkennung

Wie im Grundlagenteil beschrieben, ist es eine Kernaufgabe eines Netzwerkscanners lebendige Hosts eines Netzwerkes identifizieren zu können. Deshalb wird im ersten vergleichenden Kriterium die standardmäßige Vorgehensweise getestet, die ein Netzwerkscanner bietet, wenn es darum geht Hosts in einem Netzwerk finden zu können. Es wird vor allem überprüft wie schnell die Netzwerkscanner einen fest definierten Adressbereich untersuchen können.

### 4.3.2 Ports

Nachdem eine lebendige Adresse mit einem Host in einem Netzwerk gefunden wurde, stellt sich die Frage, wie der jeweilige Netzwerkscanner die Ports des gefunden Hostes untersuchen kann. Dabei ist die Evaluation der unterschiedlichen Scanner Ports in einem IPv6 Netzwerk scannen zu können komplexer als die der Hosterkennung.

Hier sollen die unterschiedlichen Netzwerkscanner anhand beispielhafter Port Scans aus den verschiedenen Bereichen miteinander verglichen werden. Besonders wichtig ist dabei die Genauigkeit und wie der jeweilige Netzwerkscanner den Zustand eines Ports beziffert.

### 4.3.3 Spezielle Features

Da einige Scanner über mehr Optionen verfügen, wenn es darum geht ein Netzwerk untersuchen zu können, werden spezielle Features mit aufgelistet Diese lassen sich zwar nicht auf Performance vergleichen, sind aber trotzdem ein wichtiges Kriterium, wenn es darum geht Netzwerkscanner miteinander vergleichen zu können.

### 4.3.4 Dokumentation

Ein weicher Faktor, der aber auch eine Rolle bestimmen kann, ist wie die Dokumentation des Scanners ist. Wichtig hier ist, wie ausführlich bestimmte Szenarien beschrieben werden und ob ein Nutzer sich aufgrund der Dokumentation schnell in den Scanner einarbeiten kann.

# 5 Vergleich der Netzwerkscanner

In diesem Kapitel werden die ausgewählten Netzwerkscanner anhand der vorher festgelegten Kriterien miteinander verglichen. Alle durchgeführten Scans werden dabei aus der gleichen virtuellen Maschine 4.1.2 im MWN Netz ausgeführt. Alle Versuche wurden auch am selben Tag erstellt, um jeweilige Lastunterschiede ausschließen zu können.

**Gewichtung der einzelnen Kriterien:**
Für den in 5.5 endgültigen Vergleich der Netzwerkscanner sei anzumerken das in dieser Arbeit nicht alle Kriterien gleich gewichtet werden. So nutzt ein ausführlich gut dokumentierter Netzwerkscanner wenig, wenn er die beiden wichtigeren Attribute (Host Erkennung, Ports) unzureichend aufgrund von Geschwindigkeit oder Genauigkeit erfüllt.

## 5.1 Host Erkennung

Um die Host-Erkennung der ausgewählten Scanner evaluieren zu können, werden im ersten Schritt die benötigten Befehle und Besonderheiten des Scanners zur Host Erkennung beschrieben. In 5.1 werden, die im MWN aufgenommen Zeiten, der verschiedenen Scans aufgezeigt und miteinander verglichen.

### Nmap

**Lokal** *Befehl: nmap -6 –script=targets-ipv6-multicast-echo*

Nmap unterstützt die Lokale Host Erkennung. Dabei schickt das Script ICMPv6 Echo Anfragen an die link lokale Multicast Adresse (ff02::1) um Hosts im eigenen Lokalen Netzwerk finden zu können, ohne das jeder einzige Host im Lokalen Adressbereich angepingt werden muss.

**Global:** Um den Hosts in einem anderen Netzwerk lokalisieren zu können, bietet Nmap eine Vielzahl an Möglichkeiten. In dieser Arbeit wird zuerst die standartmäßigen Optionen getestet die Nmap vorgibt, wenn man keine eigenen Optionen übergibt. Mit der standardmäßigen Host Erkennung von Nmap in globalen Netzwerken, werden viele Tests gleichzeitig ausgeführt. Um den Vergleich aber Fair zu gestalten, wird versucht Nmap mit den möglichst schnellsten Optionen laufen zu lassen. Schnellste Optionen bedeutet in diesem Fall, dass Weglassen von zusätzlicher Funktion, wie z.B der DNS Auflösung eines gefunden Hosts, welche den Test langsamer machen kann.

**Standard Host Erkennung** *Befehl: nmap -6 -sP -iL FILE*

Um Netzwerkscans in einem IPv6 Adressbereich ausführen zu können, reicht es vor dem gewünschten Befehl, ein *-6* hinzuzufügen. Dadurch kann eine Vielzahl an Befehlen, die man aus dem Scannen eines IPv4 Netzwerkes kennt, analog zu IPv6 verwendet werden. Wenn es nur gewünscht ist Hosts in einem Netzwerk zu finden, bietet sich die Option *-sn* an. Durch diese Optionen führt Nmap keine Port Scans aus, wenn es einen neuen Host in einem Netzwerk entdeckt. Ohne diese Option würde Nmap, nachdem es einen Host gefunden hat, damit beginnen standardmäßig die tausend beliebtesten Ports zu untersuchen.

Als Standard setzt sich die Option -sn ausfolgenden Proben zusammen: ICMPv6 Echo Request, TCP SYN Port 443, TCP ACK Port 80, ICMP Timestemp welches aber weggelassen wird, weil es nicht Teil von ICMPv6 ist. Insgesamt werden bei diesem Befehl für jeden einzelnen Host der zu untersuchten Adressliste, drei Ports untersucht.

**Einfacher Ping Befehl** *Befehl: nmap -6 -PE -sn -iL FILE*

Neu hier im Vergleich, zum vorherigen Befehl, ist die Option -PE diese sagt Nmap,dass nur ein einfacher ICMPv6 Echo Request benutzt werden soll. Insgesamt wird hier also nur ein einziger Port für jede Adresse untersucht.

**Einfacher Ping Befehl mit T5 Template** *Befehl: nmap -6 -PE -T5 -sn -iL FILE*

Um die Geschwindigkeit von Nmap mit denen der anderen Scanner vergleichen zu können, wird mit der Option -T5 versucht, Nmap möglichst schnell laufen zu lassen. Dabei sind die sog Timing Templates in Nmap die von T0 bis hin zu T5 gehen nur ein Syntax Sugar, weil sie eine Hand von Optionen gleichzeitig festlegen, die sonst manuell im Kommando Aufruf gesetzt werden müssten.

**Einfacher Ping Befehl mit T5 Template und ohne DNS Auflösung** *Befehl: nmap -6 -PE -T5 -sn -n -iL FILE* Um den Netzwerk Scan noch schneller gestalten zu können, bietet es sich an, die DNS Auflösung die Nmap ansonsten automatisch vornimmt abzuschalten. Dies passiert durch die Option - *n*

Tabelle 5.1: Die Zeiten für die Host Erkennung für Nmap

| **Art** | **Zeit** | **Gefundene Adressen** |
|---|---|---|
| Standard | $59m21.629s$ | 26 |
| Einfacher Ping | $19m47.769s$ | 26 |
| Einfacher Ping mit T5 | $9m58.298s$ | 26 |
| Einfacher Ping mit T5 ohne DNS Auflösung | $9m56.716s$ | 26 |
| Lokale Host Erkennung | $2.55s$ | 13 |

**Erklärung:**

In der Tabelle 5.1 sind die Daten der unterschiedlichen Befehle aufgezeigt. Es lässt sich deutlich feststellen, dass es einen großen Unterschied zwischen der standardmäßigen Host Discovery und der optimierten Einstellung zu erkennen ist.

Außerdem zeigt sich, dass im gewählten Adressraum 2001:4ca0::1-FFFF es keinen Host gibt, der nicht auf einen einfachen ICMP Echo Request antwortet, aber dafür nur auf einen TCP SYN Port 443 oder TCP ACK Port 80.

Möchte man eine 100 Prozentige Genauigkeit in dem festgelegten Adressraum haben, müsste man Proben alle Ports senden, dies ist aber mit einem zu großen Zeitaufwand verbunden. Außerdem ist es, wie in 3.2.1 beschildert unüblich das im IPv6 Spektrum Firewals ICMPv6 Echo Requests blockieren.

### Zmap

**Lokal:** Zmap unterstützt keine Lokale Host Erkennung, der Anwender müsste eine IP Adresse mit allen möglichen Lokalen IP Adressen genieren und diese testen, was zu einem zu großen Zeitaufwand führen würde.

**Global:** *Befehl: zmap -M icmp6_ echoscan –ipv6-target-file=FILE –ipv6-soure-ip=SOURCEIP*

Um globale Netzwerkscans ausführen zu können wurde das Modul icmp6_echoscan benutzt. Zmap benötigt als Input eine Liste von IPv6 Adressen und die Quell IP. Die Tabelle 5.2 zeigt die Ergebnisse dieses Netzwerkscans.

Tabelle 5.2: Die Zeiten für die Host Erkennung für Zmap

| **Art** | **Zeit** | **Gefundene Adressen** |
|---|---|---|
| icmp6_echoscan | 9,241s | 26 |

### Alive6

**Lokal** *Befehl: atk6-alive6 eth0*

Alive6 unterstützt es ein Lokales Netzwerk zu untersuchen die Zeiten befinden sich dafür in der Tabelle 5.3

**Global** *Befehl: atk6-alive6 eth0 BEREICH/FILE* Im Gegensatz zu Nmap kann der Netzwerk Scanner Alive6 mit zwei unterschiedlichen Parametern aufgerufen werden. Zum einen kann er genau wie Nmap einen File mit gültigen IPv6 Adressen als Input übergeben werden, zum anderen ist es möglich, den untersuchenden Adressraum direkt anzugeben. So untersucht der Befehl atk6-alive6 eth0 2001:4ca0::0-ffff alle Adressen angefangen von 2001:4ca0::0 bis hin zu 2001:4ca0::ffff. Ansonsten muss neben der gewählten Adresse nur noch das Interface angegeben werden. Die Tabelle 5.3 zeigt die Zeiten des Scans.

Tabelle 5.3: Die Zeiten für die Host Erkennung für Alive6

| Art | Zeit | Gefundene Adressen |
|---|---|---|
| Host Erkennung | $33,414s$ | 26 |
| Lokaler Scan | $5,867s$ | 12 |

### Scan6

**Lokal:** *Befehl: scan6 -i eth0 -L -e -v*

Scan6 unterstützt lokale Netzwerkscans. Die Zeiten befinden sich dafür in der Tabelle 5.4

**Global:** *Befehl: scan6 -t -d BERREICH / scan6 -w FILE*

Auch Scan6 kann anhand zwei unterschiedlicher Optionen ausgeführt werden, um entfernte Adressen Scannen zu können. Im ersten angehängten Befehl, wird Scan6 mithilfe der Optionen - t und -d aufgerufen. Dabei steht - t für - print-timestamp was bedeutet, dass der Scanner in der Ausgabe den genauen Zeitpunkt mitanfügen soll, nachdem eine Adresse gefunden ist.

Mithilfe der Option - d lässt sich der Zielbereich, der zu scannen ist definieren. Dabei kann dieser in der Form von 2001:4ca0::/32 oder 2001:4ca0::1-ffff übergeben werden. Standardmäßig, falls nicht weiter vorgegeben, wird ein ICMPv6 Echo Request an den ausgewählten Host geschickt und ein IPv6 Paket mit unerkannten IPv6 Optionen.

Dies wird getan da Windows Systeme nicht auf ICMPv6 Echo Request Nachrichten reagieren, die an eine Multicast Adresse gesendet werden. Der zweite Befehl mit der Option signalisiert Scan6 als Input für die zu betrachtenden Adressen, einen File einzulesen mit den zu untersuchenden Hosts. Die Zeiten für die Host Discovery mit Scan6 zeigt die Tabelle 5.4.

Tabelle 5.4: Die Zeiten für die Host Erkennung für Scan6

| Art | Zeit | Gefundene Adressen |
|---|---|---|
| Host Erkennung | $2m26.477s$ | 26 |
| Lokaler Scan | $4,062s$ | 12 |

### Chiron

**Lokal:** *Befehl: chiron_ scanner.py eth0 -mpn*

Chiron unterstützt es ein Lokales Netzwerk zu untersuchen, die Zeiten befinden sich dafür in Tabelle 5.5

**Global:** *Befehl: chiron_ scanner.py eth0 -d BEREICH -sn -threads 100* Chiron ist der einzige Scanner, der in Python geschrieben ist, um ihn zu benutzen, muss deshalb vor dem eigentlichen Scanner Python stehen. Der erste Parameter legt das Interface fest. Die Option - d definiert den gewünschten Bereich und kann genau wie Scan6 oder Alive6 Adressbereiche mit benutzen. Mit der Option -sn benutzt Chiron dieselben Mechanismen zum Scannen eines Hosts wie Nmap.

Die Besonderheit an Chiron ist die Benutzung von Threads, welche sich erheblich auf die Zeiten des Scanvorgangs auswirken können. Der Unterschied, den die Menge der Threads ausmacht, lässt sich gut in Tabelle 5.6 sehen. Hierfür wurde ein Adressbereich genommen mit 100 Adressen und es wurde Chiron anhand unterschiedlicher Thread Anzahlen aufgerufen.

Es lässt sich deutlich ein Unterschied von einem Thread hin zu 100 Threads erkennen. Außerdem ist auch ersichtlich, dass eine bloße Erhöhung der Anzahl der Threads nicht dazu führt, dass die Geschwindigkeit sich erhöht, sondern sie verringert sich sogar. Die Zeiten für Chiron sieht man in der Tabelle 5.5.

Tabelle 5.5: Die Zeiten für die Host Erkennung für Chiron

| **Art** | **Zeit** | **Gefundene Adressen** |
|---|---|---|
| Host Erkennung | $12m22,07s$ | 26 |
| Lokaler Scan | $4,201s$ | 12 |

Tabelle 5.6: Zeiten der Unterschiedlichen Threads für Chiron

| **Threads** | 1 | 10 | 100 | 150 | 200 |
|---|---|---|---|---|---|
| **Zeit** | $104,911s$ | $12,232s$ | $6,103s$ | $7,159s$ | $9,396s$ |

### Vergleich

Nachdem die Besonderheiten jedes einzelnen Netzwerkscanners in ihrem jeweiligen Unterkapitel erläutert wurden, wird in diesem Abschnitt die Zeiten der jeweiligen Netzwerkscanner miteinander verglichen. Nach Auswertung der Daten hat sich herausgestellt, dass alle Netzwerkscanner in dem untersuchenden Abschnitt, jeweils die gleiche Anzahl an Hosts, aber zu einer je unterschiedlichen Zeit gefunden haben. Für die Zeiten von Nmap, wurde dabei die schnellste Zeit mit den geringsten zusätzlichen Optionen gewählt, um die Zeiten miteinander vergleichen zu können.

**Zeit zum Scannen von 65536 Adressen:** Die Abbildung 5.1 zeigt ein Balkandiagram mit den Zeiten der unterschiedlichen Netzwerkscanner für 65536 Adressen. Diese Größe wurde der Einfachheit gewählt, weil sie den letzten Bereich (2001:4ca0::1-FFFF) des MWN Adressenraumes entspricht.

**Zeit zum Scannen von 10 Mio. Adressen:** Zur genaueren Betrachtung, wie schnell Zmap wird, wurde der Adressraum der zu untersuchenden Hosts auf 10 Mio. Adressen erhöht. Dabei wurden, die zwei schnellsten Netzwerkscanner noch einmal getestet. Die Differenz zwischen Zmap und Alive6 sieht man in der Abbildung 5.2.

Abbildung 5.1: Die Zeit zum Scannen von 65536 Adressen

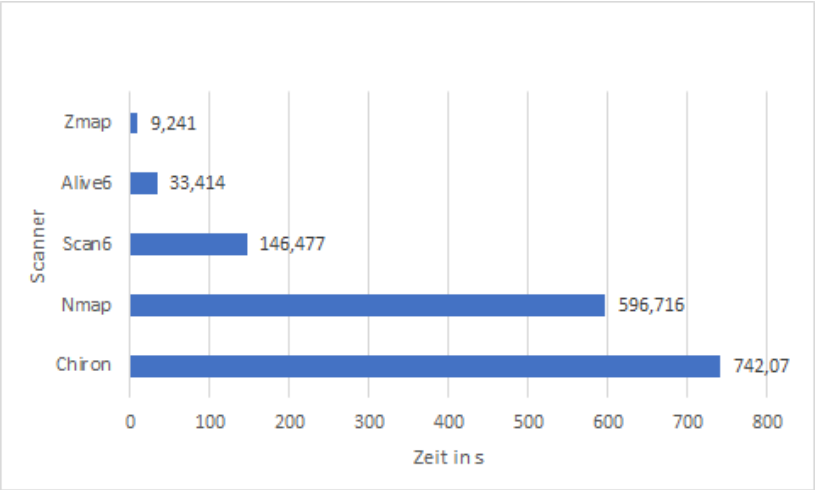
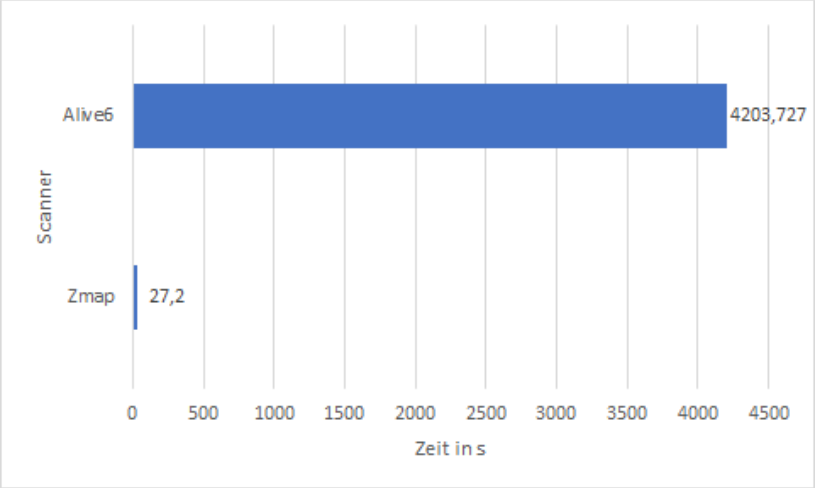



Abbildung 5.2: Die Zeit zum Scannen von 10 mio Adressen



## 5.2 Ports

Die Fähigkeit der unterschiedlichen Netzwerkscanner, die Ports einer IPv6 Adresse scannen zu können, erweist sich als schwieriger als die der Host Erkennung. Schwieriger deshalb da nach ausführen der unterschiedlichen Scanner sich gezeigt, hat dass manche nicht alle Optionen bereitstellen, die benötigt werden, um sinnvolle Port Scans machen zu können. In dem folgenden Abschnitt werden zu einzelnen Netzwerkscanner Begründungen angegeben weshalb sie sich für bestimmte Port Scans nicht eignen.

#### Anmerkungen zu einzelnen Netzwerkscannern:

**Zmap** Aufgrund der Tatsache das Zmap nur einen Port für eine beliebig große Anzahl an IPv6 Adressen untersuchen kann per Aufruf. Findet Zmap auch in vertikalen Scans keine Beachtung.

**Alive6** Alive 6 unterstützt nur TCP SYN, TCP ACK und UDP Scans. Außerdem können nur max. 255 Ports gleichzeitig gescannt werden. Aufgrund dessen, fällt dieser Scanner weg beim vertikalen Scan, da dieser hauptsächlich dazu dient alle Ports eines Hosts untersuchen zu können. Zwar könnte ein Skript geschrieben werden, welches nacheinander, diesen Scanner aufruft um auf 65536 Ports kommen zu können, aber in dem Vergleich werden die Netzwerkscanner so untersucht, wie sie für jedermann offen zugänglich sind. Auch Block Scans das heißt, mehrere Ports von mehreren Hosts können nicht mit einem Aufruf mit Alive 6 ausgeführt, werden deshalb fällt dieser Scanner für Block Scans aus.

**Scan6** Scan6 unterstützt zwar laut Dokumentation fünf Arten von TCP Scans und UDP Scans für definierte Port Bereiche, aber beim Ausführen der jeweiligen Netzwerkscans kam es zu Fehlern. So sind für UDP Scans laut Scan6 immer alle Ports offen (siehe Auflistung 5.1). Auch bei TCP Scans gibt es Probleme und es werden keine richtigen Port Zustände angezeigt (siehe Auflistung 5.2). Dem Entwickler des Scanners wurde per Mail dieser vermeintliche Bug gemeldet, es lässt sich aber auch nicht ausschließen das die Optionen zum ausführen des Scans vielleicht falsch gesetzt werden. Leider befindet sich in der Dokumentation dieses Scanners kein beispielhafter Aufruf, wie Scan6 Block Scans mit unterschiedlichen Ports ausführen kann. Bis zur Fertigstellung dieser Arbeit kam auch kein Feedback des Urhebers von Scan6. Aufgrund dieses Umstandes fällt Scan6 Was die Evaluierung der Port Scans betrifft weg.

Auflistung 5.1: Scan6 Fehlerhafter Aufruf für UDP Scans

```
1  # scan6 -i eth0 -d 2001:4ca0::1 -G udp -v -j udp:1-3
2  Target address ranges (1)
3  2001:4ca0:0:0:0:0:0:1
4  Target UDP ports: 1-3;
5  Port scan report for: 2001:4ca0::1
6  PORT STATE SERVICE
7  1/udp open tcpmux
8  2/udp open compressnet
9  3/udp open compressnet
```

Auflistung 5.2: Scan6 Fehlerhafter Aufruf für TCP Scans

```
1  # scan6 -i eth0 -d 2001:4ca0::1 -G syn -v -j tcp:1-200
2  Target address ranges (1)
3  2001:4ca0:0:0:0:0:0:1
4  Target TCP ports: 1-200;
5  Port scan report for: 2001:4ca0::1
6  PORT STATE SERVICE
```

### Welche Scanner bleiben also übrig?

Aus den eben erwähnten Problemen und Eigenschaften der Netzwerkscanner ergeben sich unterschiedliche Scan Szenarien. Für vertikale Netzwerkscans werden nur Nmap und Chiron untersucht. Da für Block Scans auch nur Nmap und Chiron übrig bleiben wird darauf verzichtet diese auszuführen. Des Weiteren unterstützt Zmap nur TCP SYN und UDP Scans, deshalb werden auch nur diese beiden für horizontale Netzwerkscans eingesetzt.

Bei der reinen Host Erkennung war es zum Durchsuchen eines Netzwerkes nicht entscheidend, ob auch möglichst viele Adressen in dem zu untersuchenden Bereich vorkamen. Beim untersuchen der unterschiedlichen Ports von Adressen ist es jedoch interessant herauszufinden, wie die einzelnen Netzwerkscanner den Status eines Portes klassifizieren, und ob alle den gleichen Zustand bestimmen. Um einfach an Möglichst viele gültige IPv6 Adressen zu kommen ,wurde deshalb mit einem Programm aus dem Toolkit von Alive6 der DNS Raum auf gültige Adressen hin untersucht. Genauere Erläuterungen zu diesem Programm befinden sich unter den speziellen Features der einzelnen Netzwerkscanner.

Durch dieses Programm konnten so über 7000 IPv6 Adressen in wenigen Sekunden gefunden werden, von denen nach Host Erkennung mit Nmap ca. 1000 online sind. Diese generierte Adresse wurde für die horizontalen Port Scans benutzt.

Im weiteren wird für den horizontalen Scan Nmap und Chiron miteinander verglichen. Diese sind die Einzigen, die alle Ports mit einem Aufruf scannen können. Block Scans werden nicht ausgeführt da für diese erstens nur Nmap und Chiron dran kommen könnten und weil Chiron eine sehr unübersichtliche Ausgabe besitzt.

**Port Evaluation Zusammengefasst:**

- Alle Port Scans, außer der vertikalen, werden anhand der generierten IP Liste erstellt (ca 1000 Adressen)
- Horizontale Port Scans werden alle an einer Adresse ausgeführt.
- Es werden bei horizontalen Scans nur TCP SYN und UDP Scans untersucht
- Scan6 kann aufgrund Fehler nicht evaluiert werden
- Block Scans fallen raus da nur Nmap und Chiron diese ausführen können und schon anhand horizontalen Scans verglichen werden

**Ausgabe der Netzwerkscanner:**

Bevor die Testergebnisse der Scanner verglichen werden, sei kurz auf die unterschiedliche Ausgabe der Netzwerk Scanner verwiesen. Alle aufgeführten Beispiele entstammen dabei des SYN Scans.

**Nmap:** Die Auflistung 5.3 zeigt beispielhaft, wie Nmap seine Ergebnisse nach durchführen des Scans ausgibt. Dabei liefert Nmap eindeutig für jede untersuchte Adresse und dort untersuchten Port den Zustand fest und den möglichen Service dahinter. Um feststellen zu können, wie viele Ports eines Horizontalen Netzwerkscans offen sind, reicht ein einfaches Grep (Kommando mit dem man in Linux einfache Textfiles durchsuchen kann und somit die Häufigkeit des Port Zustandes open aufzählen kann.)

Auflistung 5.3: Nmap Ausgabe eines geschlossenen Ports

```
1  Nmap scan report for 2001:4ca0::53:1
2  Host is up (0.00023s latency).
3  PORT STATE SERVICE
4  80/tcp closed http
5
6  #letzte Zeile eines Scans
7  Nmap done: 7148 IP addresses (1346 hosts up) scanned in 88.53 seconds
```

**Chiron:** Chiron gibt einerseits am Anfang der Ausgabe den Kompletten Scan Log aus und gegen Ende wie in der Auflistung 5.4 (Zeile 1-3) angedeutet die Adressen mit den offenen Ports. Chirons Ausgabe ist also ein JSON Objekt in der zu erst die IPv6 Adresse steht dann das Protocol dann der Port und zum Schluss das Flag. Anhand des Flags wird dann auch ermittelt ob der Port offen ist oder nicht. Falls ein Host nicht auf ein ICMPv6 Paket Antwortet wird der Port als CLOSED gesetzt, die Ausgabe dafür steht in der Auflistung 5.4

Auflistung 5.4: Chiron Ausgabe

```
1  OPENED TCP PORTS
2  ----------------
3  ['2001:4ca0:0:101::81bb:a11', ' TCP ', 'http', 'SA']
4
5  #Host der nicht auf eine ICMPv6 Anfrage antwortet
6  ['2001:4ca0:0:101::81bb:a0a', ' ICMPv6 ', 'Destination unreachable', 'Port unreachable',
7  'Target: 2001:4ca0:0:101::81bb:a0a', 'TCP port http CLOSED']
```

**Zmap:** Im Gegensatz zu Nmap liefert Zmap als Ausgabe nur eine einfache Liste mit IPv6 Adressen deren Port von Zmap als offen deklariert wurde. So gibt es für Zmap in der Ausgabe der Port Zustände nur den Zustand offen. Außerdem ist zu beachten, dass in dem vorhandenen File mit den IP Adressen auch doppelte Adressen vorhanden sind weshalb ein Benutzer zuerst alle Doppelten Adressen entfernen sollte. Danach reicht es alle gefunden Adressen zu zählen, mit einem Befehl wie wc -l File ist dies z.B möglich.

**Alive6:** Alive 6 liefert dem Benutzer als Ausgabe einen Log File aus und klassifiziert die untersuchten Hosts/ Ports nach Offen TCP RST oder geschlossen RST. Falls der Host nicht auf ein ICMPv6 Echo Anfrage antwortet wird der host als unreachable deklariert. 5.5 zeigt für alle Zustände beispielhafte Ausgaben. Am Ende des Files wird wie in Nmap eine Auflistung der untersuchten Adressen mit den gefunden Online Systemen gegeben.

Auflistung 5.5: Alive6 Ausgabe

```
1  #Beispiel eines offenen Ports
2  Alive: 2001:4ca0:10b:ff00::2:2 [80/TCP SYN-ACK]
3
4  #Beispiel eines geschlossenen Ports
5  Alive: 2001:4ca0:10c::1 [80/TCP RST]
6
7  #Beispiel eines nicht erreichbaren Hosts mit ICMPv6 Paket
8  Alive: 2001:4ca0:0:e91c::ff [ICMP host unreachable for 2001:4ca0:0:120:250:56ff:fe8f:3c86]
9
10 #Am Ende des Files
11 Scanned 7146 addresses and found 1056 systems alive
```

Zusammenfassend lässt sich sagen, dass einem Nmap viel Arbeit abnimmt und die Port Zustände eindeutig für einen Nutzer bestimmt. Ein Laie versteht die Zustände, nur wenn er mit den jeweiligen Protokollen vertraut ist und weiß was die unterschiedlichen Flags bedeuten. Im Laufe dieser Arbeit, wurden der Port Zustand mit einfachen Hilfswerkzeugen, die Linux einen anbietet, ermittelt.

Tabelle 5.7: TCP SYN Port Scan auf Port 80 anhand dnsListe

| Scanner | Zeit in s | Offen | Geschlossen | Gefiltert |
|---|---|---|---|---|
| Nmap | 88.53 | 116 | 1025 | 205 |
| Zmap | 9.198 | 116 | | |
| Alive6 | 18.710 | 109 | 858 | |
| Chiron | 107.648 | 114 | | |

#### Vergleich Horizontal SYN ACK:

Die Tabelle 5.7 zeigt die Ergebnisse des Horizontalen Tests mit Nmap, Zmap, Alive6 und Chiron. Mit den Folgenden Befehlen wurde die Scans ausgeführt.

**Ausgeführte Befehle:**

**Nmap:** *nmap -6 -T4 -sS -p 80 -n -iL FILE*

**Zmap:** *zmap -M ipv6_ tcp_ synscan -p 80 –ipv6-target-file=FILE –ipv6-source-ip=SOURCE_ IP*

**Alive6:** *atk6-alive6 eth0 -i FILE -s 80*

**Chiron:** *python chiron_ scanner.py eth0 -iL FILE -sS -p 80 -threads 100*

**Einordnung der Ergebnisse:**

**Geschwindigkeit:** Auch bei der Geschwindigkeit ergibt sich das gleiche Verhältnis ab wie bei der Host Erkennung. So ist Zmap mit 9.198 Sekunden der schnellste Netzwerkscanner, wobei der eigentliche Scan noch schneller ablaufen würde, aber Zmap einen Default cooldown Wert besitzt. Das heißt, Zmap wartet eine bestimmte festgelegte Zeit, bevor es terminiert, auch wenn schon alle Ports durchsucht worden sind. Danach folgt Alive6 mit 18.710 Sekunden. Wie bei der Host Erkennung ist Nmap auf dem vorletzten Platz und Chiron auf dem letzten.

**Genauigkeit:** Wenn es um das Port scannen geht, bemisst die Genauigkeit einen höheren Stellenwert, als die der reinen Geschwindigkeit. Nmap und Zmap finden am meisten offene Ports (116). Außerdem teilt Nmap die Portzustände weiter in geschlossen und gefiltert ein.

Interessant bei dieser Messung ist es, weshalb Alive6 und Chiron auf weniger offene Ports kommen. Um das herauszufinden, wurden alle gefunden IP Adressen indem der Port 80 offen ist als eigener File herausgenommen.

Dieser File wurde dann verwendet, um mit Alive6 und Chiron noch einmal den Netzwerkscan durchführen zu können. Dabei ist aufgefallen das Alive6 für 7 Adressen keinen TCP SYN Ack, als Flag setzt, sondern NDP ausgibt. Siehe Auflistung 5.6 All diese Adressen befinden sich im gleichen lokalen Netzwerk, wie die Versuchsmaschine aus diesen Grund wurden die Ports nicht als offen angezeigt.

Des Weiteren hat sich bei Chiron gezeigt das beim Ausführen der Testes zwei IP Adressen nicht im Log waren, nach einzelnem Testen der IP Adressen konnte aber der richtige Zustand bestimmt werden. Der genaue Grund hierführ könnte ein Bug sein, indem Chiron bei einer größeren Anzahl an IPv6 Adressen vielleicht manche übersieht. Ähnlich wie Scan6 und Alive6 ist Chiron ein Scanner, der hauptsächlich von einer Person betrieben und gewartet wird.

Auflistung 5.6: Alive 6 Adressen mit NDP Flag

```
1 Alive: 2001:4ca0:0:110::81bb:fb01 [NDP 00:50:56:8f:9e:18]
2 Alive: 2001:4ca0:0:110::81bb:fb0a [NDP 00:50:56:8f:01:43]
3 Alive: 2001:4ca0:0:110::81bb:fb0f [NDP 00:50:56:8f:1d:d2]
4 Alive: 2001:4ca0:0:110::81bb:fb11 [NDP 00:50:56:8f:38:a4]
5 Alive: 2001:4ca0:0:110::81bb:fb14 [NDP 00:50:56:8f:96:77]
6 Alive: 2001:4ca0:0:110::81bb:fb15 [NDP 00:50:56:8f:39:34]
7 Alive: 2001:4ca0:0:110::81bb:fb16 [NDP 00:50:56:8f:75:95]
```

Tabelle 5.8: UDP Port Scan auf Port 80 anhand dnsListe

| Scanner | Zeit in s | Offen | Geschlossen | Gefiltert | Offen\|Gefiltert |
|---|---|---|---|---|---|
| Nmap | 79.86 | 0 | 779 | 183 | 384 |
| Zmap | 9,219 | 0 | | | |
| Alive6 | 18,412 | 0 | | | |
| Chiron | 104,053 | 0 | | | |

### Vergleich Horizontal UDP

Siehe Tabelle 5.8.

**Ausgeführte Befehle:**

**Nmap:** *nmap -6 -T4 -sU -p 80 -n -iL FILE*

**Zmap:** *zmap -M ipv6_udp –ipv6-target-file=FILE –ipv6-source-ip=SOURCE_IP*

**Alive6:** *atk6-alive6 eth0 -u FILE -s 80*

**Chiron:** *python chiron_scanner.py eth0 -iL FILE -sU -p 80 -threads 100*

**Geschwindigkeit:** Die Geschwindigkeit ist Analog zum vorherigen Vergleich.

**Genauigkeit:** Über die Genauigkeit lässt sich wenig sagen, außer das Nmap der einzige Netzwerkscanner ist der UDP Port Zustände noch genauer definiert.

Tabelle 5.9: Vergleich Nmap und Chiron

| Scanner | Typ | Zeit in s | Offen | Gesch | Gefiltert | Offen\|Gefil | Ungefiltert |
|---|---|---|---|---|---|---|---|
| *Nmap* | | | | | | | |
| | SYN | 1683.32 | 2 | 65534 | | | |
| | UDP | 13427.80 | 2 | 65529 | | 5 | |
| | ACK | 1622.95 | | | | | 65536 |
| | FIN | 1677.84 | | 65534 | | 2 | |
| | Xmas | 1824.65 | | 65534 | | 2 | |
| | Null | 1661.54 | | 65534 | | 2 | |
| *Chiron* | SYN | 344.467 | 2 | | | | |
| | UDP | 177.788 | | | | | |
| | ACK | 353.641 | | | | | |
| | FIN | 353.997 | | | | | |
| | Xmas | 350.484 | | | | | |
| | Null | 353.997 | | | | | |

### Vergleich Vertikal Nmap und Chiron

Für diesen Vergleich, wurde ein Host hergenommen, der gleiche für Nmap und Chiron und es wurden alle Ports mit jeweils sechs unterschiedlichen Port Scan Arten ausgewählt. Die Tabelle 5.9 zeigt die Ergebnisse. Wie in den anderen Vergleichen wird der Scan aufgrund von Geschwindigkeit und Genauigkeit beurteilt.

**Ausgeführte Befehle:**

**Nmap:** *nmap -6 -sn -T4 -p0-65535 ZIEL*

**Chiron:** *chiron_scanner.py eth0 -d ZIEL -sS -p 0-65535*

**Geschwindigkeit:** Bei der Geschwindigkeit gibt es, gravierende Unterschiede zwischen den beiden Netzwerkscannern Chiron und Nmap. Für die fünf unterschiedlichen TCP Scans braucht Nmap im Durchschnitt 1698,1 Sekunden. Chiron hingegen nur 351.32 Sekunden. Chiron ist also fast fünfmal schneller als Nmap. Im UDP Bereich ist Chiron mit 177.788 Sekunden, sogar 75-mal Schneller als Nmap.

**Genauigkeit:** Bei der Genauigkeit muss beachtet werden, dass Chiron nicht bestimmen kann, ob ein Port gefiltert ist oder nicht. Wie Nmap Port zustände deklariert wurde ausführlich im Grundlagenteil erläutert. Chiron legt den Zustand der Ports anhand der Flags fest, die zurückgesendet werden. So bedeutet SYN-ACK das ein Port offen ist und RESET-ACK oder ICMPv6 Destination unreachable heißt der Port ist geschlossen.

Für den Benutzer werden nur offene Ports am Ende angezeigt. Die geschlossenen Port zustände, muss sich der Benutzer anhand der gesetzten Werte in der Ausgabe von Chiron selbst zusammenzählen. Chiron ist also was die Port Ausgabe angeht noch nicht ausgereift. Als

Beispiel des Problems der Ausgabe von Chiron im Vergleich zu Nmap wird der UDP Scan im Detail betrachtet. Die Auflistung 5.7 zeigt das Ergebnis für Nmap und die Auflistung 5.8 die für Chiron. Bei der Ausgabe für Chiron, wurde extra der Abschnitt gewählt der von Nmap als offener Port 123 (ntp) anzeigt wurde. Anhand dieses Vergleichs lässt sich deutlich erkennen das wenn Chiron am Ende der Ausgabe nicht die Offenen Ports aufführt es schwer zu bestimmen ist ob die Ports offen sind. Deshalb ist in der Tabelle auch nur eindeutig für den SYN Scan zwei offene Ports von Chiron eingetragen, da aus dem Log File nicht genau gesagt werden ob ein Port geschlossen ist oder nicht.

Auflistung 5.7: Ausgabe des UDP Scans von Nmap

```
Not shown: 65529 closed ports
PORT STATE SERVICE
0/udp open|filtered unknown
123/udp open ntp
161/udp open snmp
496/udp open|filtered pim-rp-disc
547/udp open|filtered dhcpv6-server
2029/udp open|filtered unknown
48347/udp open|filtered unknown
```

Auflistung 5.8: Ausgabe des UDP Scans von Chiron

```
['2001:4ca0::1', ' UDP ', 'ntp', '2001:4ca0::1', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'ICMPv6']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
['2001:4ca0:0:110::81bb:fb17', 'UDP']
```

Aufgrund des beschriebenen Umstands punktet Nmap in der Genauigkeit vor allem, wenn ein Benutzer auch die gefilterten Ports bestimmen möchte. Hier würde es sich anbieten, Chiron zu erweitern und eine bessere und genauere Port Ausgabe zu entwickeln. Diese hätte den Vorteil, schneller horizontale Netzwerkscans mit gleichbleibender Genauigkeit ausführen zu können.

### Vergleich

Bei der Evaluation der Fähigkeit der IPv6 Netzwerkscanner Port zu untersuchen punktet Zmap mit seiner Geschwindigkeit und Nmap mit seiner Genauigkeit. Nmap ist der einzige Netzwerkscanner der Ports nicht nur danach definiert, ob sie offen oder geschlossen sind und kann somit mehr Information aus einem horizontalen Scan, ableiten als z.B Chiron. Die Tabelle 5.10 zeigt die unterstützen Port Scans der einzelnen Netzwerkscanner. Bis auf Scan6 würden auch alle Scan Arten der Netzwerkscanner überprüft.

Tabelle 5.10: Die Auflistung der unterschiedlichen Port Scan Methoden

| Scanner | SYN | CON | ACK | FIN | XMAS | NULL | UDP |
|---|---|---|---|---|---|---|---|
| Nmap | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Zmap | ✓ | | | | | | ✓ |
| Alive6 | ✓ | | ✓ | | | | ✓ |
| Scan6 | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| Chiron | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |

## 5.3 Auswahl Spezieller Features

Neben der Host Erkennung und dem Scannen von Ports, hat sich bei Evaluierung der Netzwerkscanner, gezeigt, dass manche Scanner auch weitere nützliche Methoden aufweisen, die für Netzwerkbetreiber eines Netzwerkes von großer Nützlichkeit wären. Genau aus diesem Grund sind zusätzliche Funktionen ebenso ein Kriterium beim Vergleich der Netzwerkscanner. Im folgenden wurde für jeden Netzwerkscanner die jeweiligen Features aufgelistet und kurz erläutert.

#### Nmap:

Nmap besitzt viele weitere nützliche Möglichkeiten, Hosts scannen zu können. Mithilfe der Nmap Scripting Engine (NSE) ist es Benutzern möglich Nmap durch Skripte zu erweitern.

Auch für IPv6 gibt es nützliche Skripte. Des Weiteren liefert Nmap schon aus dem IPv4 Bereich sinnvolle Möglichkeiten, Geräte eines Netzwerkes untersuchen zu können. Im weiteren Verlauf der Arbeit sind deshalb einige nützliche Features rund um die Schachstellenanalyse eines IPv6 Adressenbereiches aufgegliedert.

Außerdem befinden sich im Anhang zu jeder beispielhaften Ausführung des jeweiligen Aufrufes. Komplizierte Funktionen, wie die der Dienst Erkennung werden, im Rahmen dieser Arbeit nichts in Detail besprochen, wichtig ist hier nur, diese als Feature für den jeweilig untersuchten Scanner aufzuführen mit einer kurzen Beschreibung.

**Dienst und Versionserkennung:** *Befehl: nmap -6 -A -T4 ZIEL*

Allein zu wissen ob ein Port offen ist, reicht einem Angreifer oder einem Schwachstellentester noch nicht aus um mögliche potenzielle Schwächen ausfindig machen zu können. In dem Buch [Lyo09] bringt der Autor Gordon Fyodor Lyon als Beispiel, das oft Inhaber eines Webservers mehrere Webserver auf einen Port haben können (Test, Stageing) neben dem des typischen Portes, der für Port 80 reserviert ist. Oder es kann auch sein,dass der Betreiber eines Servers sich nicht an typische Konventionen halten und ihre Dienste auf andren Ports laufen.

So schreibt Gordon “People can and do run services on strange ports.” Außerdem führt der Autor aus, dass selbst wenn die standardmäßigen Dienste auf ihren normalen Ports laufen, es wichtig ist, welche Version der jeweiligen Dienste die Anfrage auf diesem Port entgegen nehmen. Genau deshalb bietet Nmap die Dienst und Versionserkennung, an welche auch für IPv6 voll funktionsfähig ist. „IPv6 is supported, including TCP, UDP, and SSL over TCP.”.

**Betriebssystem-Erkennung:** *Befehl: nmap -6 -O -T4 ZIEL*

Die Fähigkeit das Betriebssystem eines Zielrechners erkennen zu können ist ein nützliches Feature von Nmap. In gleicher Weise kann es gerade bei der Inventarisierung eines Netzwerkes von enormen Vorteil sein, wenn die gefunden Geräte anhand ihres Betriebssystem eingeteilt werden können. Außerdem liefert das Betriebssystem weitere mögliche Schwachstellen so macht es einen großen Unterschied, ob sich hinter der Adresse ein Drucker oder ein Windows Desktop befindet. Einfach gesagt nutzt Nmap eine Datenbank mit Heuristiken, die Aufgrund unterschiedlicher Antworten zu TCP/IP Proben das Betriebssystem ermitteln.

**Spezielle Scripte extra für IPv6:**

**targets-ipv6-map4to6** Befehl:
*nmap -6 –script targets-ipv6-map4to6 –script-args newtargets,targets-ipv6-map4to6.IPv4Hosts=129.187.0.0/16,targets-ipv6-subnet=2001:4ca0::/32*

Dieses Skript wird in der Pre-Scan-Phase ausgeführt, um IPv4-Adressen auf IPv6-Netzwerke abzubilden und in die Scan-Warteschlange aufzunehmen. Die Technik ist allgemeiner als das, was technisch als ÏPv4-mapped IPv6-Adressen"bezeichnet wird. Die unteren 4 Bytes der IPv6-Netzwerkadresse werden durch die 4 Bytes der IPv4-Adresse ersetzt. Wenn das IPv6-Netzwerk ::ffff:0:0:0/96 ist, dann erzeugt das Skript IPv4-mapped IPv6-Adressen. Wenn das Netzwerk ::/96 ist, dann erzeugt es IPv4 kompatible IPv6-Adressen.

**targets-ipv6-wordlist** *nmap -6 –script targets-ipv6-wordlist –script-args newtargets,targets-ipv6-subnet=2001:4ca0::/32*

Fügt IPv6-Adressen zur Scanwarteschlange hinzu, indem er eine Wortliste mit hexadezimalen "Wörtern"verwendet, die Adressen in einem bestimmten Subnetz bilden.

### Zmap:

Zmap besitzt kein spezielles Features

### Scan6:

Einer der Autoren von [GC16] hat auch Scan6 entwickelt und hat aus diesen Erkenntnissen einige Skripte seinem Netzwerkscanner hinzugefügt welche grob gesagt, als Ziel hatten alle einen kleineren Adressenraum zu definieren, um IPv6 Adressen in einem Netzwerk finden zu können.

**Virtuelle Maschinen** *scan 6 -d 2001:4ca0::/32 –tgt-virtual-machines all –ipv4-host 129.187.0.0/16*

Scannen Sie nach virtuellen Maschinen (sowohl VirtualBox als auch vmware) im Präfix 2001:4ca0::/32. Die zusätzliche Informationen über das vom Hostsystem verwendete IPv4-Präfix werden genutzt, um Folgendes zu erreichen den Suchraum zu reduzieren.

**Low Byte Adressen** *scan6 -d 2001:4ca0::0-500:0-1000*

Einfache möglichkeit um die untersten Adressen einer IPv6 Adresse durchsuchen zu können.

**Embedded IPv6 Adressen** *scan6 -d 2001:4ca0::/32 –tgt-ipv4 ipv4-32 –ipv4-host 129.187.0.0/16*
Suche nach IPv6-Adressen des Netzwerks 2001:4ca0::/32, die das IPv4-Präfix einbetten. 129.187.0.0/16 (mit der 32-Bit-Kodierung).

**Embedded Port Adressen** *scan6 -d 2001:4ca0::/32 -g*
Durchsucht Port Adressen in einem gewünschten Berreich

### Alive6:

Alive6 entstammt einem Toolkit welches neben dem Scanner viele Nützliche kleinere Programe hat häufig zum Angriff eines IPv6 Netzwerkes. Als ein Nützliches Features welches direkt in Alive6 liegt könnte man ipv4/range bezeichnen.

**IPv4 Berreich** *atk6-alive6 -4 129.187.0.0/16 eth0 2001:4ca0::/32* Mit dieser Option kann wie aus dem RFC in [GC16] beschrieben werden versucht den Adressraum zu verkleinern indem man nach embeded IPv4 Adressen sucht.

**dnsrevenum6** *atk6-dnsrevenum6 DNSSERVER IPv6NETWORK* Dnsrevenum ist eine interessante Möglichkeit, an IPv6 Adressen und Subnetze über DNS zu gelangen. In [GC16] wird diese Technik beschrieben die ursprünglich von einem anderen Autor in einem Blog Post verfasst wurde, welcher nicht mehr online ist. Bei dieser Technik wird die ip6.arpa Zone durchgegangen und es wird nach PTR Rekords gesehen. Es wird anhand der zu Untersuchenden Adresse überprüft, ob der DNS Server, der ausgewählt wurde, irgendeinen Eintrag von diesem Adressbereich hat. Wenn der DNS Server einen Eintrag hat, wird er (no error ) zurückgeben falls nicht NXDOMAIN. So kann schnell der ganze Baum abgegangen werden, um Adressen aus dem DNS Server zu finden.

### Chiron:

Chiron ist mehr als nur ein einfacher Netzwerkscanner sondern bietet auch mehrere Module um den Adressbereich eines Netzwerkes anzugreifen, da diese aber für diese Arbeit nicht von Bedeutung sind werden sie nur kurz aufgelistet um zu zeigen das Chiron viele interessante Module besitzt.

**Smart Scan:** *chiron_ scanner.py eth0 -sM -pr 2001:4ca0::1 -pr 64 -iC TEXTFILE* Ähnlich wie der Word Scan in Nmap nur das es bei Chiron möglich ist eigene Wortkombinationen zu erstellen und Chiron zu übergeben. (hatte beim Testen leider einen Fehler)

**Sniffer:** *python chiron_ scanner.py eth0 -rec -stimeout 20* Das Lokale Netz wird belauscht *-stimeout* legt fest für wie lange gelauscht wird

### Vergleich:

Einige Netzwerkscanner beinhalten viele interessante Funktionen. Viele von diesen Funktionen sind darauf ausgelegt anhand von Mustern der Adressraum des IPv6 Bereichs zu verkleinern. Nmap bietet aber mit seinen Versionen und Diensterkennung und Betriebssystem-Erkennung das nützlichste Feature.

## 5.4 Dokumentation

Von der Gewichtung her ist die Dokumentation vielleicht das geringste Kriterium weshalb ein Nutzer den jeweiligen Netzwerkscanner benutzten sollte um sich einen Überblick über sein IPv6 Netzwerk zu verschaffen.

**Nmap:** Nmap ist der älteste Netzwerkscanner, der untersuchenden Scanner, deshalb ist Nmap auch sehr ausführlich Dokumentiert und besitzt auf ihrer Webseite ein gute Dokumentation welche auch als Buch [Lyo09] herauskam. Welche besonders beim Erstellen dieser Arbeit äußerst hilfreich war, um sich in die Thematik des Netzwerkscannes einlesen zu können.

Nur ist das Buch gerade was IPv6 betrifft ziemlich veraltet und beinhaltet auch wenig spezifische Information zu IPv6. Nmap unterstützt IPV6 support seid 2002 doch viele Features kamen erst mit der Zeit. So steht auch auf der offiziellen Webseite und im Buch noch:

> Der TCP-Connect-Scan ist der standardmäßig eingestellte TCP-Scan-Typ, falls der SYN-Scan nicht möglich ist

Das ist dann der Fall, wenn der Benutzer kein Recht hat, rohe Pakete zu senden, oder wenn er IPv6-Netzwerke scannt.". In der offizielen Dokumentation stehst also das man mit Nmap keinen SYN Scan machen kann, aber seid Version 6 unterstützt Nmap raw Paket port Scanning. Dies ist nur ein Beleg wie veraltet die Dokumentation gerade hinsicht auf das IPV6 scannen ist.

**Zmap:** Zmap für Ipv4 besitzt ein eigenes Wikipedia und viele Paper welche die Architektur von Zmap auflisteten oder Zmap mit anderen Netzwerkscanner vergleichen. Für die Ipv6 Module gibt es nur ein readme auf dem Github Projekt indem kurz die einzelnen Module aufgelistet sind.

**Scan6:** Offizielle Dokumentation befindet sich im eigenen Gifthub Projekt [fgo] . Außerdem hat der Autor des Scanners zusammen mit Tim Chown auch ein eigenes RFC [GC16] herausgebracht, welche auch einige Techniken des Scanners näher vorstellt. Es sind zwar einige Beispielaufrufe für Befehle vorhanden, aber leider keine für das Port scannen.

**Alive6:** Für Alive6 gibt es nach Recherche des Autor keine offizielle Dokumentation das open Source Projekt liegt auf Github in [VT19]. Außerdem gibt es auf [kal] so etwas wie eine Dokumentation in dem das Toolkit kurz beschrieben wird.

**Chiron:** Im Github Projekt von Chiron [Aat18] befindet sich ein ausführliche Dokumentation zu den verschiedenen Features des Scanners mit Beispielen.

**Vergleich:**

Eine gute Dokumentation geht auf jede einzelne Option und Kombination ein und gibt dem Nutzer darüber hinaus auch ein gutes Technisches Verständnis über die verschiedenen Scans eines Netzwerkscanners. Bei der Evaluation der einzelnen Scanner, was die reine Dokumentation angeht haben sich Nmap und Chiron als Gewinner herausgestellt. Die Dokumentation

in Nmap geht auch sehr allgemein auf technisches Verständnis ein, welches für den Umgang mit Netzwerkscannern benötigt wird. Auch die Dokumentation in Chiron ist ausführlich und weit mehr als eine kurze Beschreibung aller benötigten Optionen. Auf den dritten Platz Scan6. Auf den vorletzten Platz Alive6 und auf den letzten Platz Zmap was die reine IPv6 Dokumentation betrifft.

## 5.5 Gesamtvergleich

Die Tabelle 5.11 zeigt in diesem Kapitel ausgewählte Features im Vergleich. Dabei wurde aufgrund der Evaluation jeder Scanner in den einzelnen Kategorien miteinander verglichen. Je mehr *XXX* ein Netzwerkscanner in dieser Kategorie hat desto besser ist er. Falls nichts in dem Feld steht, unterstützt dieser Netzwerkscanner die ausgewählte Kategorie nicht.Ein - bedeutet der Netzwerkscanner unterstützt laut Dokumentation die Kategorie, aber sie konnte nicht getestet werden. Insgesamt ist Nmap in allen belangen besser als seine Konkurenten, bis auf die Geschwindigkeit hier überragt Zmap mit seiner besonderen Architektur.

Tabelle 5.11: Überblick über die Evaluierten Netzwerkscanner

| | Nmap | Zmap | Alive6 | Scan6 | Chiron |
|---|---|---|---|---|---|
| Local Host Discovery | X | | X | X | X |
| Remote Host Discovery | X | XXX | XX | XX | X |
| Musster Adressen | XX | | X | XX | |
| TCP SYN | XX | XX | X | - | XX |
| TCP CON | XX | | | | |
| TCP ACK | XX | | X | - | X |
| TCP FIN | XX | | | - | X |
| TCP XMAS | XX | | | - | X |
| TCP NULL | XX | | | - | X |
| UDP | XX | XX | X | - | X |
| Horizontale Scans | XX | | | - | X |
| Vertikale Scans | X | XXX | | - | |
| Block Scans | XXX | | | - | X |
| Versions / OS Erkennung | XXX | | | | |
| Dokumentation | XXX | X | X | X | XX |

# 6 Praktikabilität der Netzwerkscans

Der Hintergrund dieser Arbeit ist das Fehlen einer aktuellen Überwachung für das IPv6 Netz des MWN. So soll ähnlich, wie für IPv4 in Zukunft auch ein Überblick über angeschlossene Geräte im IPv6 Netzwerk verschafft werden können.

Als ersten Schritt für ein Monitoring dieses Netzes, dient diese Arbeit, indem sie geeignete IPv6 Netzwerkscanner vorstellt und miteinander vergleicht. Aufgrund der gewonnen Erfahrungen mit dem Umgang von aktuellen IPv6 fähigen Netzwerkscannern lassen sich erste Ansätze für das überwachen eines IPv6 Netzwerkes entwickeln.

In diesem Kapitel soll kurz skizziert werden wie die in Kapitel 5 evaluierten Netzwerkscanner benutzt werden können, um ein IPv6 Netzwerk untersuchen zu können.

Folgende fragen gilt es zu klären, wenn ein Betreiber eines Netzwerkes seine angeschlossenen Geräte untersuchen will:

1. Wie kann der Betreiber sich eine Übersicht über alle seine angeschlossenen Geräte beschaffen?
2. Wie kann ein Betreiber einzelne Ports eines gefunden Hosts scannen und auf mögliche Schwachstellen untersuchen?
3. Wie lange würde es theoretisch dauern bestimme Adressbereiche zu untersuchen?
4. Wie lange würde es dauern alle Ports mehrerer Adressen untersuchen zu können?

**1. Wie kann der Betreiber sich eine Übersicht über alle seine angeschlossenen Geräte beschaffen?**

Hierbei gibt es zwei Möglichkeiten einmal, können sehr schnell und einfach Lokale Netzwerk Scans ausgeführt werden, aber dies würde heißen das es in jedem Lokalen Netzwerk ein Gerät gibt, welche diese Ausführt und die Ergebnisse dann weiter leitet. Extern und Global würde das bedeuten das verschiedene Netzwerkscanner einen Bereich untersuchen würden bei großen Netzwerken kann dies aber sehr lange dauern.

Aber es ist Prinzipiell möglich den gesamten Adressraum untersuchen zu können. Eine Andere Möglichkeit, die aufgrund des großen Netzbereiches in IPv6 auftritt, ist den Adressraum anhand ausgewählter Muster (low byte, IPv4 embedded, ... ) kleiner zu machen und ihn dann zu untersuchen. Wie effizient diese Möglichkeit ist, hängt von der Adressiersungspolitik des jeweiligen Betreibers ab.

Außerdem würden Muster zwar die Arbeit des Netzwerkbetreibers vereinfachen aber dafür könnten auch Angreifer leichter an Potentielle Opfer kommen. Hinzu kommt auch das gar nicht geklärt ist wie IPv6 Adressen ermittelt worden sind die sich alle paar Stunden eine neue Adresse geben.

**2. Wie kann ein Betreiber einzelne Ports eines gefunden Hosts scannen und auf mögliche Schwachstellen untersuchen?**

Wenn der Host identifiziert ist, bietet ein Port Scanner wie Nmap alle Werkzeuge an um diesen auf Ports und weiteren Schachstellen untersuchen zu können. Außerdem ist es mit einem Netzwerkscanner wie Zmapv6 möglich sehr schnell Horizontale Port Scans über einen großen Adressraum ausführen zu können.

Mit etwas Arbeit könnte auch ein Block Scan effizient mit Zmapv6 ausgeführt werden. Für genaue Netzwerk Scans und um besser den jeweiligen Zustand des Ports unterscheiden zu können und Schwachstellen herausziehen bietet sich Nmap an. Darüber hinaus ist in dieser Arbeit Nmap der einzige Netzwerkscanner mit dem es im IPv6 Adressbereich möglich ist die potenzielle Schwachstellen herauszubekommen, sprich welcher konkreter Dienst mit welcher Version hinter einem Port läuft.

**3. Wie lange würde es theoretisch dauern bestimme Adressbereiche zu untersuchen?**

Für die Berechnung der theoretischen Zeit wird vom Adressierungsplan des MWN ausgegangen. Der gesamte globale Adressierungsbereich des MWN ist 2001:4ca0::/32. Jedem Kunden wird aus diesem Block ein /48 Netz zugeordnet. Daraus ergibt sich, dass ein Kunde 65536 Netze zur Verfügung hat, die jeweils $2^64$ Geräte haben können.

Aufgrund dieser Ausgangslage sind zum einen die theoretischen Zeiten interessant für den gesamten MWN Raum und einmal die für ein explizites Netzwerk, mit dem jeweiligen Subnetz. Die Tabelle 6.1 zeigt die Hochgerechneten Zeiten für die Host-Erkennung. Die einzelnen Zeiten basieren auf die für 65536 Aufgenommen Adressen und bei Zmapv6 und Alive6 auf die für 10 Mio. durchsuchten Adressen.

In der Tabelle lässt sich deutlich erkennen, dass selbst der mit Abstand schnellste Netzwerkscanner in keinen endlichen Zeiten einen Adressraum von 2 96 oder 2 64 scannen kann. Einzig ein Adressraum von $2^{32}$ scheint mit 3,3 Stunden noch passabel. Dies ist auch der Gesamtadressraum, den ein IPv4 Netzwerk hat.

Tabelle 6.1: Die Hochgerechnete Zeiten für Host Erkennung

| Adressgröße | **Chiron** | **Nmap** | **Scan6** | **Alive6** | **Zmap** |
|---|---|---|---|---|---|
| 1 | $0,01132s$ | $0,00911s$ | $0,00224s$ | $0,00042s$ | $0,00001s$ |
| $2^{16}$ | $12,37min$ | $9,95min$ | $2,44min$ | $33,41s$ | $9,241s$ |
| $10 * 10^6$ | $31,45h$ | $25,30h$ | $6,21h$ | $70,06min$ | $27,2s$ |
| $2^{24}$ | $52,77h$ | $42,43h$ | $10,41h$ | $1,96h$ | $46,47s$ |
| $2^{32}$ | $1,54y$ | $1,24y$ | $111,11d$ | $20,90d$ | $3,30h$ |
| $2^{64}$ | 6,62 $*10^9$ y | $5,33 * 10^9y$ | $1,31 * 10^9y$ | $2,46 * 10^8y$ | $1,62 * 10^6y$ |
| $2^{80}$ | $4,34 * 10^{14}y$ | $3,49 * 10^{14}y$ | $8,57 * 10^{13}y$ | $1,61 * 10^{13}y$ | $1,06 * 10^{11}y$ |
| $2^{96}$ | $8,97 * 10^{26}y$ | $7,21 * 10^{26}$ | $1,77 * 10^{26}y$ | $3,33 * 10^{25}y$ | $2,19 * 10^{23}$ |

**4. Wie lange würde es dauern alle Ports mehrerer Adressen untersuchen zu können?**

Sobald die Liste der lebendigen IPv6 Hosts in einem Netzwerk ermittelt wurde, ist es wichtig, Ports untersuchen zu können und mögliche Schwachstellen ausfindig zu machen. Die Tabelle 6.2 zeigt so beispielhaft wie lange es für eine bestimmte Anzahl an Hosts dauern würde, um mit einem TCP SYN Scan alle Ports überprüfen zu können. Ausgehen von der gemessenen Zeit der Netzwerkscanner, die sie brauchen, um alle Ports eines Hosts zu scannen, wurden die restlichen Zeiten hochgerechnet.

Für einen horizontalen Port Scan eigenen sich zum jetzigen Stand nur Nmap und Chiron. Es ist deutlich zu erkennen, dass gerade bei steigender Anzahl von Adressen, in der Größe Chiron sehr viel schneller ist als Nmap. So braucht Chiron bei einer Hostanzahl von 100000 Adressen vier Jahre weniger als Nmap. Aber wie in der Evaluation der beiden Netzwerkscanner bereits gezeigt, ist die Ausgabe und das Erkennen in Chiron eher mangelhaft. Daher würde sich Chiron bei jetzigem Stand nur dazu empfehlen offene TCP Ports abzusuchen.

Es würde sich auch anbieten bei dieser hohen Zeit Blockscans mit Zmap zu simulieren oder ein eigenes Modul dafür schrieben, welche eine Reihe an Ports als Parameter übergeben werden kann. Eine weitere Möglichkeit, die bestehen würde, wäre es keine Block Scans mit der vollständigen Anzahl an Ports auszuführen, sondern nur die kritischen oder die beliebtesten. Dies würde den Scanvorgang erheblich kürzen, auch wenn dafür einbüßen bei der Genauigkeit gemacht werden müssten.

Tabelle 6.2: Die Hochgerechnete Zeiten für TCP SYN Port Scan mit allen Ports

| **Hostanzahl** | **Nmap** | **Chiron** |
|---|---|---|
| 1 | $1683,32s$ | $344,47s$ |
| 10 | $280,55s$ | $57,41s$ |
| 100 | $46,76h$ | $9,57h$ |
| 1000 | $19,48d$ | $3,99d$ |
| 10000 | $194,83d$ | $39,87d$ |
| 100000 | $5,34y$ | $1,10y$ |

**Anmerkung: Möglichkeit nach Mustern zu suchen:**

Eine weitere Möglichkeit, den großen Adressraum eines IPv6 Netzwerkes zu verkleinern, ist es Muster zu benutzen beim Durchsuchen der Hosts. Im Rahmen dieser Bachelorarbeit stellte der Autor fest, dass beim Durchsuchen des IPv6 Adressraums des MWN, manchmal eingebettete IPv4 Adressen aufgekommen sind.

Das heißt, konkret, dass die ersten zwei Stellen der IPv4 Adresse des MWN (129.187) als Hexadezimalzahlen umgewandelt worden sind. Die Hexadezimalzahl von 129 ist 81 und die von 187 ist bb. Als Beispiel für diese These betrachten wir die IPv6 und IPv4 Adresse der Versuchsmaschine.

**2001:4ca0:0:110::81bb:fb17** und **129.187.251.23**

Wandelt man nun die Stellen 129 187 251 und 23 einzeln als Hexadezimalzahl um ergibt sich 81 bb fb 17.Für die Erstellung der IPv6 Adresse der Versuchsmaschine wurde also die schon vorhandene IPv4 Adresse eingebettet.

Dies ist ein erstes Indiz, das für einige Adressen eingebettete IPv4 Adressen benutzt wurden. Da aber die Verwendung von Mustern für das eigene Netzwerk auch sicherheitskritische Aspekte bereithalten würden, sobald ein Angreifer davon weiß könnte er diese Muster benutzten und leicht potentielle Angriffsziele finden. Es wurde in dieser Arbeit nicht überprüft, ob konkrete Muster, wie sie beschrieben worden sind auch am MWN angewendet worden sind. Diese Information wäre aber für den Betreiber des Netzwerkes wichtig und würde es ihn erlauben einen Bereich von $2^{32}$ Adressen in unter vier Stunden (mit Zmap) untersuchen zu können.

# 7 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurden IPv6 Netzwerkscanner am MWN evaluiert. Um die Scanner vergleichen zu können, wurde zuerst benötigte Grundlagen in Kapitel 2 und 3 erläutert. Danach wurden Kriterien zum Vergleich der Netzwerkscanner und der Versuchsaufbau in Kapitel 4 festgelegt. Am Ende der Arbeit wurden die in Kapitel 5 gewonnen Ergebnisse der Evaluation dazu benutzt, um die Effektivität der Netzwerkscans in einem großen IPv6 Netzwerk, am Beispiel des MWN in Kapitel 6 überprüft.

Für die zu Beginn der Arbeit gestellten Forschungsfragen ergeben sich folgende Resultate:

Ein wichtiges Kriterium für einen Netzwerkscanner, ist die Fähigkeit schnell und genau Hosts in einem festgelegten Adressraum zu finden. Nur das finden von Hosts reicht nicht aus, um weitere sicherheitskritische Merkmale des Gerätes zu erfassen. Deshalb stellt neben der Identifikation eines Hosts auch die Untersuchung der Ports ein wichtiges Kriterium da. Neben diesen beiden festen Säulen eines Netzwerkscanners, sind auch weitere von besonderer Relevanz. So ist es wichtig ob ein Netzwerkscanner weitere Hilfsmöglichkeiten anbieten um Adressen untersuchen zu können. Der unwichtigste Faktor, gerade für die Benutzung des jeweiligen Netzwerkscanners, ist die Frage wie gut der jeweilige Scanner dokumentiert ist und ob ein Benutzer sich schnell mit ihm vertraut machen kann.

Die Netzwerkscanner unterscheiden sich sehr deutlich von einander. Zum einem weisen die evaluierten Scanner sehr große Unterschiede, auf was ihre reine Geschwindigkeit angeht. So ist Zmapv6 der mit Abstand schnellste Netzwerkscanner. Dagegen ist Nmap deutlich langsamer, aber ermöglichst es sehr genau die Port Zustände eines Hosts zu bestimmen. Nmap benutzt auch ein breites Spektrum um die Zustände der Ports anzugeben. Bei den anderen Netzwerkscannern war die Ausgabe oft mangelhaft. Es war daher auch sehr schwer festzustellen ob ein Netzwerkscanner wie Chiron den Zustand eins Ports richtig bestimmt. Die wichtigsten zusätzlichen Features hat wohl Nmap, mit dem es möglich ist die Version eines Dienstes oder das Betriebssystem eines Gerätes zu bestimmen. Aufgrund der immensen Adressgröße von IPv6 haben auch viele Netzwerkscanner Hilfsmittel um nach gängigen Mustern in Adressen zu suchen. Bei der Dokumentation punktet Nmap auch wenn es für IPv6 veraltet ist und viele der damals aufgeführten Inhalte nicht mehr stimmen.

Es hat sich bei der Evaluation der Netzwerkscanner und beim hochrechnen der Zeiten für Adressgrößen die ein IPv6 Netzwerk wie das des MWN hat, gezeigt das es nicht möglich ist den gesamten IPv6 Adressraum effizient scannen zu können. Aber es konnte gezeigt werden das Nmap was die weitere Untersuchung eines Ports betrifft alle gängigen Werkzeuge unterstützt wie für IPv4. Das Problem bei IPv6 ist es also den gesamten Adressraum zu untersuchen. Die Arbeit konnte aber auch zeigen das ein Netzwerkscanner wie Zmapv6 den ge-

samten Adressraum von $2^{32}$ effizient untersuchen kann. Es ist durchaus möglich einen kleinen Teilbereich eines Netzwerkes zu untersuchen. Darüber hinaus benutzt viele Netzwerkscanner Muster, mit denen der durchsuchte Raum noch weiter eingeschränkt werden kann. Als Fazit sei gesagt das es möglich ist den IPv6 Adressraum zu untersuchen und das es wahrscheinlich einfach ist feste Adressen also oftmals Server in einem Adressraum zu finden sobald man ein Gefühl für den Adressierungsplan des Netzwerkes hat. Was IPv6 Netzwerkscanner aber nicht können, ist jedes Gerät in einem Netz zu erfassen, da vor allem die meisten "normalen"Hosts sprich Nutzer oft ihre Adresse ändern und nur temporäre globale Adressen benutzen.

### Ausblick

Aufgrund der beschriebenen Probleme sei als Ausblick auf eine andere Arbeit verwiesen, die sich damit auseinandergesetzt hat, wie ein großes Netzwerk sich einen Überblick über seine Hosts machen könnte. Die Arbeit von den Autoren Matěj Grégr, Petr Matoušek und Co beschäftigt sich in [GMPv01] mit einem praktischen Monitoring Ansatz, um das Netzwerk einer Universität überwachen zu können. Im folgenden sei der Ansatz der Autoren kurz skizziert: Die Autoren benutzen ein System mit dem sie jeden Benutzer eindeutig anhand eines Tupels (IPv6, MAC, IPv4, Login name) identifizieren.

Die Informationen für die jeweiligen Attribute dieses Tupels bekommen sie aus den unterschiedlichsten Quellen. Die Tabelle 7.1 entstammt genau aus ihrer Arbeit und verdeutlicht die unterschiedlichen Möglichkeiten die benutzt werden, um einen Benutzer eindeutig identifizieren zu können. Anhand von Netflow Rekords (kurz ist eine Möglichkeit, bei der ein Gerät Informationen über einen IP Datenstrom innerhalb des Gerätes per UDP nach außen exportiert). Wird ermittelt, ob ein Benutzer seine IPv6 Adresse ändert und die Informationen an einer zentralen Stelle gesammelt und in einer Datenbank exportiert.

Dabei entsteht auch eine große Menge an Daten, dass auch zu einem Problem werden kann, wie die Autoren in der Arbeit erläutern. Die Architektur zur Überwachung eines IPv6 Netzwerkes in dieser Arbeit, ist es zentral in einem Netzwerk durch, Netflow Rekords Informationen über IPv6 Adressen zu gewinnen und diese an einer weiteren Stelle zu versenden, wo sie gebündelt werden und am Ende so ein Nutzer des Netzwerkes eindeutig identifiziert werden kann.

Tabelle 7.1: Data input for the central monitoring system (Quelle [GMPv01])

| **OSI layer** | **Data** | **Source Information** |
|---|---|---|
| L2 | RADIUS log (using 802.1x) | Login, MAC address |
| L2 | Switching table | Switch port, MAC address |
| L3 | Router Neighbor Cache | IPv6 address, MAC address |
| L3 | Router ARP table | IPv4 address, MAC address |
| L4-7 | Netflow records | IPv4/IPv6 addresses, ports |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[Aat18] Aatlasis: aatlasis/Chiron. https://github.com/aatlasis/Chiron. Version: Aug 2018. – abgerufen am 11. Juni 2019

[ARCJ11] Amante, Shane ; Rajahalme, Jarno ; Carpenter, Brian E. ; Jiang, Sheng: IPv6 Flow Label Specification. RFC 6437. http://dx.doi.org/10.17487/RFC6437. Version: November 2011 (Request for Comments)

[BHFV18] Borgolte, Kevin ; Hao, Shuang ; Fiebig, Tobias ; Vigna, Giovanni: Enumerating Active IPv6 Hosts for Large-Scale Security Scans via DNSSEC-Signed Reverse Zones. In: 2018 IEEE Symposium on Security and Privacy (SP) (2018). http://dx.doi.org/10.1109/sp.2018.00027. – DOI 10.1109/sp.2018.00027

[BLRS19] Bailey Lee, Cynthia ; Roedel, Chris ; Silenok, Elena: Detection and characterization of port scan attacks. (2019), 06

[Bra89] Braden, Robert T.: Requirements for Internet Hosts - Communication Layers. RFC 1122. http://dx.doi.org/10.17487/RFC1122. Version: Oktober 1989 (Request for Comments)

[DH06] Deering, Dr. Steve E. ; Hinden, Bob: IP Version 6 Addressing Architecture. RFC 4291. http://dx.doi.org/10.17487/RFC4291. Version: Februar 2006 (Request for Comments)

[DH17] Deering, Dr. Steve E. ; Hinden, Bob: Internet Protocol, Version 6 (IPv6) Specification. RFC 8200. http://dx.doi.org/10.17487/RFC8200. Version: Juli 2017 (Request for Comments)

[DM07] Davies, Elwyn B. ; Mohácsi, János: Recommendations for Filtering ICMPv6 Messages in Firewalls. RFC 4890. http://dx.doi.org/10.17487/RFC4890. Version: Mai 2007 (Request for Comments)

[fgo] fgont/ipv6toolkit. https://github.com/fgont/ipv6toolkit/blob/master/manuals/scan6.1. – abgerufen am 11. Juni 2019

[GC06] Gupta, Mukesh ; Conta, Alex: Internet Control Message Protocol (ICMPv6) for the Internet Protoc RFC 4443. http://dx.doi.org/10.17487/RFC4443. Version: März 2006 (Request for Comments)

[GC16] Gont, Fernando ; Chown, Tim: Network Reconnaissance in IPv6 Networks. RFC 7707. http://dx.doi.org/10.17487/RFC7707. Version: März 2016 (Request for Comments)

[GMPv01] Grégr, Matěj ; Matoušek, Petr ; Podermański, Tomáš ; Švéda, Miroslav: Practical IPv6 Monitoring on Campus Best Practice Document Produced by the CESNET-led Working Group on Network Monitoring (CESNET BPD 132). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.369.3424. Version: May 2001

[goo18] google: Google ipv6 statistics. https://www.google.com/intl/en/ipv6/statistics.html. Version: 2018. – abgerufen am 11. Juni 2019

[GSGC16] Gasser, Oliver ; Scheitle, Quirin ; Gebhard, Sebastian ; Carle, Georg: Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. In: Proc. of 8th Int. Workshop on Traffic Monitoring and Analysis. Louvain-la-Neuve, Belgium, April 2016

[HR16] Helmut Reiser, Stefan M.: Das Münchner Wissenschaftsnetz (MWN) Konzepte, Dienste, Infrastruktur und Management. https://www.lrz.de/services/netz/mwn-netzkonzept/mwn-netzkonzept-2017.pdfl. Version: 2016. – abgerufen am 24.03.2019

[kal] THC-IPV6. https://tools.kali.org/information-gathering/thc-ipv6. – abgerufen am 11. Juni 2019

[KK10] Kawamura, Seiichi ; Kawashima, Masanobu: A Recommendation for IPv6 Address Text Representation. RFC 5952. http://dx.doi.org/10.17487/RFC5952. Version: August 2010 (Request for Comments)

[Lyo09] Lyon, Gordon F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. USA : Insecure, 2009. – ISBN 0979958717, 9780979958717

[NJT07] Narten, Dr. T. ; Jinmei, Tatsuya ; Thomson, Dr. S.: IPv6 Stateless Address Autoconfiguration. RFC 4862. http://dx.doi.org/10.17487/RFC4862. Version: September 2007 (Request for Comments)

[Ped19] Pediditis, Nikolas: Handling IPv4 Runout. https://ripe78.ripe.net/presentations/72-RS-Feedback_Final.pdf. Version: 2019. – abgerufen am 11. Juni 2019

[PVL+03] Perkins, Charles E. ; Volz, Bernie ; Lemon, Ted ; Carney, Michael ; Bound, Jim: Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315. http://dx.doi.org/10.17487/RFC3315. Version: Juli 2003 (Request for Comments)

[si6] https://www.si6networks.com/tools/ipv6toolkit/

[VT19] Vanhauser-Thc: vanhauser-thc/thc-ipv6. https://github.com/vanhauser-thc/thc-ipv6. Version: May 2019. – abgerufen am 11. Juni 2019

[ZD13] Zakir Durumeric, J. Alex H. Eric Wustrow W. Eric Wustrow: ZMap: Fast Internet-wide Scanning and Its Security Applications. In: Proceedings of the 22nd USENIX Security Symposium (2013), aug