anexia

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Route Injection

Projekt T3_2000

für die Prüfung zum

Bachelor of Science

des Studienganges Informationstechnik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Leon Louis Schoch

Abgabedatum 18. September 2023

Bearbeitungszeitraum	27 Wochen
Matrikelnummer	1015290
Kurs	TINF21B5
Ausbildungsfirma	Anexia Deutschland GmbH Karlsruhe
Betreuer der Ausbildungsfirma	Stephan Peijnik-Steinwender (B.Sc.)
Gutachter der Studienakademie	Prof. Dr. Markus Strand

Erklärung

Ich versichere hiermit, dass ich meine Projekt T3_2000 mit dem Thema: Route Injection selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum Unterschrift

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung vom Dualen Partner vorliegt.

Zusammenfassung

Ein Distributed Denial of Service (DDoS)-Angriff kann eine starke Auslastung der betroffenen Systeme verursachen. Dies kann einen Absturz zur Folge haben, oder den Zugriff auf die Systeme verhindern. Um dieses Problem zu lösen wurde der Route Injection Service entwickelt, mit welchem ein Nutzer in der Lage ist, Netzwerkroute über Border Gateway Protocol (BGP)-Communities zu manipulieren. Ein DDoS-Angriff kann daher in ein Blackhole geroutet, und eine Belastung der Zielsysteme verhindert werden.

A DDoS-Attack can cause a high load on the attacked systems. As a result, the systems might be inaccessible or crash. To solve this problem, we developed the route injection service, which enables a user to manipulate network routes via BGP-Communities. A DDoS-Attack can then be routed into a blackhole, and a strain on the target systems can be avoided.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Code Snippets

Formelverzeichnis

Abkürzungsverzeichnis

Kapitel 1

Einleitung

Die zunehmende Abhängigkeit von digitalen Kommunikationsnetzwerken und die kontinuierliche Weiterentwicklung der globalen Infrastruktur haben zu einer signifikanten Steigerung des Datenverkehrs im Internet geführt. Während diese Fortschritte zahlreiche Vorteile für die Gesellschaft mit sich bringen, eröffnen sie auch neue Herausforderungen im Hinblick auf die Sicherheit und Stabilität des Netzwerkbetriebs. In diesem Zusammenhang gewinnt die Fähigkeit, den Datenverkehr effektiv zu leiten und gleichzeitig gegen potenzielle Bedrohungen zu schützen, zunehmend an Bedeutung. Das BGP, als das fundamentalste Routing-Protokoll im Internet, spielt eine kritische Rolle bei der Bestimmung der optimalen Routen für den Datenverkehr zwischen Autonomen Systemen (ASen). Allerdings hat die BGP-Protokollsuite bisher nur begrenzte Möglichkeiten zur gezielten Beeinflussung des Datenverkehrs in Ausnahmesituationen oder bei Sicherheitsvorfällen geboten. Eine solche Ausnahmesituation tritt beispielsweise auf, wenn ein Netzwerkressourcen-Engpass aufweist oder wenn bösartige Akteure versuchen, den Datenverkehr abzufangen oder zu manipulieren. Die vorliegende Forschung widmet sich daher der Entwicklung eines innovativen Ansatzes, der es ermöglicht, Internet-Routen über BGP gezielt in sogenannte „Blackholes“zu lenken. Dieses Konzept zielt darauf ab, den Datenverkehr von bestimmten Quellen oder zu bestimmten Zielen hinzuleiten, indem die betreffenden Routen im Netzwerk auf Blackholes abgebildet werden. Diese Blackholes repräsentieren Pfade im Netzwerk, die keinen tatsächlichen Datenaustausch ermöglichen, sondern den Verkehr effektiv abfangen und isolieren. Durch die Einrichtung dieser Blackholes wird eine maßgeschneiderte Methode zur Verteidigung gegen DDoS-Angriffe sowie zur effizienten Nutzung von Ressourcen in Überlastsituationen geschaffen. Die Motivation für dieses Projekt liegt darin,

die Flexibilität und die Sicherheitsaspekte von BGP-Routings zu erweitern, um den heutigen Anforderungen an die Netzwerksicherheit und -stabilität gerecht zu werden. Durch die Schaffung eines Mechanismus zur Blackhole-Routing kann das Risiko von Datenverkehrsumleitung durch bösartige Einflüsse minimiert und die Möglichkeit zur gezielten Netzwerkressourcenlenkung maximiert werden. Die Ergebnisse dieses Projekts haben das Potenzial, die bestehenden Ansätze zur Netzwerkverwaltung und -sicherheit zu erweitern und somit einen bedeutenden Beitrag zur Aufrechterhaltung der Integrität und Effizienz globaler Kommunikationsnetzwerke zu leisten.

Kapitel 2

Grundlagen

2.1 Einführung in die Problematik

Um im Falle eines DDoS Angriffs schnell reagieren zu können, muss es eine bequeme und einfache Möglichkeit geben, Routen zu manipulieren. Hierfür wurde das Projekt Remote Triggered Blackholing gestartet. Im Falle eines DDoS-Angriffs könnten somit IP Präfixe des Angreifers gezielt in ein Blackhole geroutet werden. Eine Belastung der Zielsysteme könnte somit verhindert werden, da die boshaften Pakete des Angreifers somit nicht beim Zielsystem ankommen würden, sondern in das schwarze Loch (Blackhole) weitergeleitet werden. Um die Routen in Routern manipulieren zu können, müssen diese über Injektoren in die Router injiziert werden. Im Verlaufe dieser Projektarbeit wird die Entwicklung der Injektoren Komponente und der Aufbau einer Staging(Testing) Umgebung genauer dargelegt. Der Aufbau und die Entwicklung der API Komponente wurde bereits zu einem großteil in der T1000 erläutert, jedoch wurde im Rahmen der T2000 diese um einen Delete-Endpunkt erweitert. [Schoch 2022]

2.2 Technologie Selektion

2.2.1 Django Rest Framework

„Django ist ein Web-Framework, dessen Ziel es ist, die Entwicklung von Web Applikationen schnell, einfach und übersichtlich zu machen. Das Django Representational State Transfer (REST) Framework, hier nachfolgend als Django Rest Framework (DRF) bezeichnet, ist ein REST Framework welches auf Django basiert. Mit DRF lassen sich REST-ful APIs schnell und einfach gestalten. Hierfür bietet Django eine Reihe an vorgefertigten Hilfestellung an, welche im Verlaufe dieser Projektarbeit näher erläutert werden. Datenbankmodelle werden hier einfach programmatisch deklariert und anschließend von Django automatisch verwaltet. Über Objekte können somit einzelne Werte aus der Datenbank entnommen werden, ohne sich mühsam mit Structured Query Language (SQL) Queries auseinandersetzen zu müssen. Sowohl Django als auch DRF basieren auf der Programmiersprache Python.“ [Vgl. Schoch 2022, S. 8]

2.2.2 Hashicorp Consul

„Consul, entwickelt von Hashicorp, ist eine Netzwerk Service Lösung, welche eine sichere Kommunikation zwischen Services und Applikation erlaubt. Consul kann sowohl redundant mit mehreren Nodes, als auch standalone genutzt werden. Für diese Projektarbeit, wird eine standalone Lösung eingesetzt und es wird lediglich die Key-Value Store Funktion genutzt. Mit dieser Funktion können Key-Value [...] Paare über das Netzwerk in Consul gespeichert werden.“ [Schoch 2022]

2.2.3 Docker

Docker ist Platform zur Containerisierung von Anwendungen. Hierdurch wird die Möglichkeit geschaffen eine isoliertes und leichtgewichtige Umgebung zu schaffen, welche sonst lediglich mittels Virtuellen Maschinen (VMs) möglich wäre. Durch Docker wird auf produktiven System durch die zusätzliche Isolationsschicht der Containerisierung eine weitere Sicherheitsstufe hinzugefügt, welche potenziellen Angreifern den Zugriff auf das Hostsystem erschwert.

2.2.4 Bird

Der Bird Internet Routing Daemon (Bird) ist eine Open-Source-Routing-Software, die als Router fungiert. Bird implementiert unter anderem BGP,

um Routing-Informationen zwischen Routern auszutauschen und optimale Routenentscheidungen zu treffen. Bird arbeitet neben anderen BGP-Routern, um BGP-Sessions aufzubauen, Routing-Updates auszutauschen und Routing-Informationen zu speichern. Bird kann BGP-Routen exportieren und an andere Router weitergeben, indem es BGP-'Update'-Messages verwendet und Exportregeln in seiner Konfigurationsdatei folgt. Diese Regeln definieren, welche Routen exportiert werden sollen und können durch Filter und Richtlinien gesteuert werden. Durch den Export von BGP-Routen ermöglicht Bird eine effiziente und zuverlässige Kommunikation und Weiterleitung in großen Netzwerken.

2.3 Stand der Technik

Das BGP ist ein Protokoll des Internet-Routings, das die *besten* Wege für den Datenverkehr zwischen ASen bestimmt. Im ursprünglichen Sinne war mit einem AS eine Organisation mit einem Standort gemeint, welche intern über ein internes routing Protokoll verfügte. Mit der Zeit hat sich die Bedeutung eines AS abgewandelt und eine Autonome System Nummer (ASN) kann von einer Organisation Standortübergreifend verwendet werden bzw. eine Organisation kann über mehrere ASNs verfügen. Es verwendet Peering-Verbindungen zwischen Routern, um Informationen über erreichbare Netzwerke auszutauschen und die optimalen Pfade für den Datenaustausch zu ermitteln. Anders als bei herkömmlichen Routing Protokollen wie dem Routing Information Protocol (RIP) oder Open Shortest Path First (OSPF), wird hier eine direkte Transmission Control Protocol (TCP) Verbindung zwischen Routern(Neighbours/Nachbarn) hergestellt. Eine weitere Unterscheidung besteht darin, dass es sich bei BGP um 'Policy'-basiertes Routing, im Vergleich zu 'Metrik' basierten Routing handelt. Konkret bedeutet dies, dass ein AS selbst bestimmen kann, wie Datenverkehr geroutet werden soll, sollte das AS über mindestens zwei Uplinks verfügen.

Wenn zwei BGP Nachbarn eine TCP Verbindung aufgebaut haben, beginnen diese BGP Informationen in Form von Nachrichten auszutauschen. Jede Nachricht besteht aus einem Header, und dem tatsächlichen Inhalt. [Vgl. Beijnum 2002, S. 19 f.] Um eine BGP Verbindung herzustellen, müssen sich Router über eine 'Open'-Message verbinden. Diese wird direkt nach dem Aufbau der TCP Verbindung ausgetauscht. [Vgl. Beijnum 2002, S. 20 f.]

Version	My AS	Hold time	Identifier	Parlen	Optional parameters
1 byte	2 bytes	2 bytes	4 bytes	1 byte	0-255 bytes

Tabelle 2.1: Aufbau der 'Open'-Message
Quelle: [RFC4271 Rekhter, Hares und Li 2006] in Anlehnung an [Beijnum 2002, S. 20]

Sollte die Open-Message erfolgreich vom Gegenstück angenommen worden sein, sendet dieser eine 'Keepalive'-Message zurück. Anschließend wird die BGP-Routentabelle über 'Update'-Messages ausgetauscht. [Vgl. Beijnum 2002, S. 20]

UR length	Withdrawn routes	PA length	Path attributes	NLRI
2 bytes	Variable	2 bytes	Variable	Variable

Tabelle 2.2: Aufbau der 'Update'-Message
Quelle: [RFC4271 Rekhter, Hares und Li 2006] in Anlehnung an [Beijnum 2002, S. 20]

Durch die 'Update'-Message werden die eigentlichen Informationen übertragen. Hierdurch können neue Routen hinzugefügt, oder alte Routen zurückgezogen werden. Ein nicht optionales Attribute ist der 'AS_PATH', welcher beschreibt, über welche AS bestimmte Präfixe zu erreichen sind.

BGP-Communities sind ein Mechanismus, mit welchem Netzwerkbetreiber spezifische Gruppen oder Kategorien von Präfixen markieren können. Diese Markierungen, als „Communities“ bezeichnet, können verwendet werden, um Routen zu identifizieren und zu beeinflussen, wie sie von anderen ASen interpretiert werden. Durch die Verwendung von Communities können Netzwerkbetreiber das Routing auf feinere Weise steuern und anpassen, ohne die Kernstruktur des BGP-Netzwerks zu verändern. Die Manipulation von Routen mittels BGP Communities erfolgt, indem einem bestimmten Präfix eine oder mehrere BGP-Communities zugewiesen werden. Andere AS können dann diese Community-Markierungen verwenden, um spezifische Aktionen auszuführen, wie z.B.:

- Pfadwahl beeinflussen: Durch das Zuweisen von Communities zu bestimmten Präfixen können Netzwerkbetreiber festlegen, wie andere AS ihre Routen interpretieren sollen. Dies kann dazu verwendet werden, den bevorzugten Weg für den Datenverkehr zu beeinflussen.
- Traffic-Engineering: Netzwerkbetreiber können Communities verwenden, um den Datenverkehrsfluss zu steuern. Durch Markieren von Präfixen können sie bestimmte AS dazu anleiten, den Datenverkehr auf bestimmten Wegen zu leiten, um Netzwerkressourcen effizienter zu nutzen.
- Blackhole-Routing: BGP Communities können dazu genutzt werden, bestimmte Präfixe zu markieren und den Datenverkehr über Blackholes zu lenken, um Angriffe oder Überlastungen zu bewältigen. Spezielle für Blackholing wurde eine eigene Community definiert: `65535:666` [Vgl. King u. a. 2016]
- Routenfilterung: AS können Community-Markierungen verwenden, um präzise Routenfilterung durchzuführen. Damit können sie bestimmte Routen von bestimmten Quellen oder für bestimmte Zwecke filtern oder akzeptieren.

Die Verwendung von BGP Communities ermöglicht eine flexiblere und zielgerichtete Steuerung des Internet-Routings. Netzwerkbetreiber können so gezielt auf unterschiedliche Anforderungen reagieren und gleichzeitig die Integrität und Stabilität des BGP-Netzwerks aufrechterhalten.

Kapitel 3

Architekur

Die Architektur des Route Injection Service besteht aus drei wesentlichen Bestandteilen, welche entweder direkt verbunden sind oder mittels Hashicorp Consul Daten austauschen können.

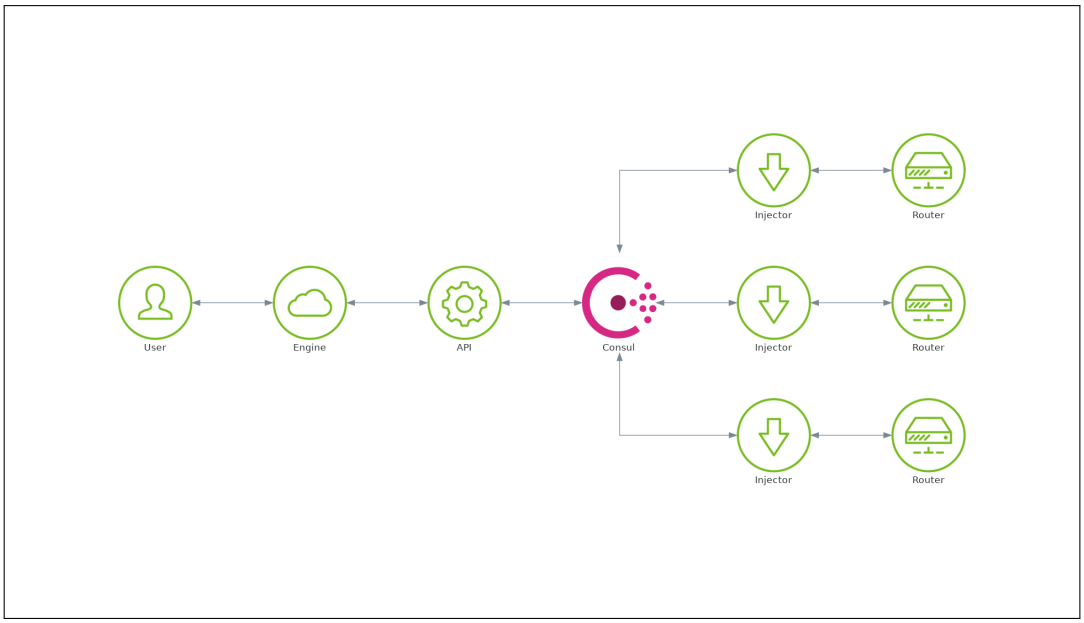

Abbildung 3.1: Route Injection Architektur
Quelle: Firmenintern

3.1 API

Die API ist dafür verantwortlich die Eingaben des Users, welche über die Engine übermittelt wurden zu überprüfen und zu validieren. Sind die Eingaben nicht korrekt, so gibt die API eine entsprechende Fehlermeldung zurück. In der Zukunft wird die API auch dafür verantwortlich sein entsprechende Monitoring Endpunkte zur Verfügung zu stellen, sodass der allgemeine Status des Service überwacht werden kann.

3.2 Hashicorp Consul

Hashicorp Consul, im weiteren Verlauf nur ‘Consul‘ genannt, wird als Zwischenspeicher für Routen und deren injizierte BGP-Communities verwendet. Des Weiteren können Injectoren hier Ihren ‘Heartbeat‘ abspeichern.

3.3 Injector

Der Injector bezieht periodisch(alle 5 Sekunden) die in Consul gespeicherten Routen. Sollte es hier eine Änderung gegeben haben, wird eine Konfigurationsdatei für den Bird Routingdaemon neu erstellt. Anschließend wird über das ’Bird Controlsocket’ der Befehl zum Neuladen der Konfiguration gegeben.

3.4 Router

Als Router wird der Bird Routingdaemon eingesetzt. Dieser stellt eine BGP-Session mit einem physischen Router her, welcher die von Bird zu Verfügung gestellten Router importiert und innerhalb des BGP-Netzwerks weitergibt.

Kapitel 4

API Komponente

4.1 Aufgaben

Die API ist die Schnittstelle des Service und außen stehenden Technologien wie der Anexia Engine. Ihre Hauptaufgabe besteht darin, eine strukturierte Interaktionsmöglichkeit zu bieten, die es internen Benutzern über Systeme wie der Anexia Engine ermöglicht, BGP-Routen mit zugehörigen BGP-Communities in das Netzwerk zu injizieren. Dies geschieht durch die Annahme von JavaScript Object Notation (JSON)-Anfragen, die spezifische Informationen enthalten, nämlich IPv4- oder IPv6-Präfixe und die entsprechenden BGP-Communities. Die API führt eine umfassende Validierung der eingehenden Daten durch, um sicherzustellen, dass die bereitgestellten Informationen korrekt und im erwarteten Format vorliegen. Diese Validierung umfasst die Überprüfung der Richtigkeit der IP-Adressbereiche sowie die syntaktische Korrektheit der zugeordneten BGP-Communities. Durch diesen Schritt wird gewährleistet, dass nur gültige Informationen in das System eingebracht werden. Die validierten Daten werden anschließend an Consul, über dessen eigene API übermittelt. Die Daten werden so abgelegt, dass der Injector einen erleichterten Zugriff hat.

4.2 Umsetzung

Da die Konzeption und Implementierung der API schon umfassend in der Projektarbeit T1000 erläutert wurde, wird auf eine Wiederholung dessen verzichtet. In diesem Bericht wird lediglich die Implementierung des ‘Delete‘-Endpunkts dargestellt, da dieser aus zeitlichen Gründen nicht mehr in den ersten beiden Praxisphase implementiert werden konnte, jedoch ein Grundbestandteil des entwickelten Service ist.

Die Implementierung eines ‘Delete‘-Endpunkts in der API, mittels des Django Rest Frameworks, ermöglicht das Löschen von Routen aus dem System.

```
1  class BaseRouteViewSet(
2      CreateModelMixin,
3      ReadOnlyModelViewSet,
4      BaseRequestViewSet,
5  ):
6      @action(detail=False, url_path=r"([A-Za-z-_/]*)status/(?P<
       task_info_id>[0-9a-z-]+)")
7      def status(self, request, task_info_id):
8          route_object = get_object_or_404(
9              self.serializer_class.Meta.model, task_info_id=
       task_info_id
10         )
11         propagate_status(route_object)
12         return super().status(request, task_info_id)
```

Code Snippet 4.1: BaseRouteViewset Klasse

Der in Snippet 4.1 gezeigte Code stellt eine Mutterklasse dar, von welcher sowohl der ‘Create‘, als auch ‘Delete‘-Endpunkt erben. Durch diese Klasse wird die Möglichkeit gegeben, von der Anexia Engine erwartete Endpunkte einfach zu implementieren, ohne dass sich ein Entwickler mit den Feinheiten dessen auseinandersetzen muss. Da hier die `CreateModelMixin` Klasse geerbt wird, stellt sich das DRF automatisch ein ’POST’-Requests für diesen Endpunkt zu akzeptieren.

```
class DeleteRouteViewSet(BaseRouteViewSet):
    queryset = DeleteRoute.objects.all()
    serializer_class = DeleteRouteSerializer

    def perform_create(self, serializer):
        super().perform_create(serializer)
        delete_route(serializer.instance)
```

Code Snippet 4.2: DeleteRouteViewset Klasse

Die tatsächliche Implementierung fällt durch das Erben von der 'BaseRouteViewSet' Mutterklasse sehr simpel aus. Durch das Überschreiben der `perform_create` Methode, welche vom DRF zur Verfügung gestellt wird, kann diese als Hook benutzt werden um eigenen Code ausführen zu lassen. Mit der Super Methode wird sichergestellt, dass die nicht überschriebene Ursprungsmethode von `perform_create` ausgeführt wird. Das DRF erstellt dann einen Datenbankeintrag mit den vom Nutzer eingegeben Werten. Vor dem Ende des Kontextes der Methode wird noch eine weitere Methode `delete_route` aufgerufen.

```
def delete_route(instance):
    consul_instance = prepare_consul(os.getenv("CONSUL_HOST"),
    os.getenv("CONSUL_PORT"))
    prefix = str(instance.prefix)
    prefix_encoding = get_prefix_encoding(prefix)
    consul_instance.kv.delete(
        f'v1/route/global/{prefix_encoding}/{prefix.replace
    ("/", "_")}'
    )
    update_active_injectors(instance)
```

Code Snippet 4.3: delete_route Methode

Hier findet nun das eigentliche Übermitteln der Daten an Consul statt.

Kapitel 5

Injector Komponente

5.1 Aufgaben

Der Injector ist der zentrale Baustein des Route Injection Service, der die Möglichkeit bietet, mittels BGP Communities, Routen in das Netzwerk zu injizieren. Der Injector erfüllt dabei eine Reihe von wesentlichen Aufgaben:

Zuallererst ist der Injector für die Konvertierung der von der API empfangenen Routen in eine für den Router (Bird) verständliche Konfigurationsdatei verantwortlich. Diese Konvertierung ist von entscheidender Bedeutung, um die Weiterleitung der Routen an den Router in einem kompatiblen Format sicherzustellen. Während die Validierung der Präfixe und Communities von der API Komponente übernommen wird, hat der Injector eine eigene Validierung für Routen, welchen über den Emergency-Mode angegeben werden, da hier die API Komponente überbrückt wird. Bei auftretenden Konflikten oder Unstimmigkeiten kann der Injector angemessene Maßnahmen ergreifen, um die Integrität der anderen Komponenten und schlussendlich des Netzwerks, zu gewährleisten. Ein wichtiger Aspekt ist auch die aktive Kommunikation des Injectors mit dem Router (Bird). Diese Kommunikation erfolgt, um die generierten Konfigurationsänderungen effektiv zu übertragen und sicherzustellen, dass die injizierten Routen nahtlos in das Routing-Protokoll des Routers integriert werden. Schließlich stellt der Injector durch präzises loggen sicher, dass im Falle eines Fehlers, oder im schlimmsten Fall, bei einem Absturz der Komponente, Ereignisse festgehalten werden. Zusammenfassend fungiert der Injector als entscheidende Schnittstelle, die die Funktionen der API und des Routers miteinander verbindet. Mit seiner intelligenten Konvertierung und Verwaltung

von Routen durch BGP Communities gewährleistet er, dass die gewünschten Routing-Änderungen präzise und effizient im BGP-Netzwerk implementiert werden.

5.2 Umsetzung

5.2.1 Generieren der Config Files für Bird

Um die Routen an den Bird Routing Daemon übermitteln zu können, müssen diese erst in eine für Bird verständliche Konfigurationsdatei umgewandelt werden.

Integrität der Konfigurationsdatei sicherstellen

5.2.2 Status der Routen von Bird abfragen

Evaluation der pybird Bibliothek

5.2.3 Bird und Bird6 aufteilen

5.2.4 Realisierung des Heartbeats

5.2.5 Emergency-Mode implementieren

5.3 Testen

Kapitel 6

Staging Umgebung

6.1 Planung

6.2 Umsetzung

Kapitel 7

Fazit

Literatur

Beijnum, Iljitsch van [2002]. *Building Reliable Networks with the Border Gateway Protocol*. O'Reilly [siehe S. 13, 14].

King, Thomas u. a. [Okt. 2016]. *BLACKHOLE Community*. RFC 7999. doi: 10.17487/RFC7999. url: https://www.rfc-editor.org/info/rfc7999 [siehe S. 15].

Rekhter, Yakov, Susan Hares und Tony Li [Jan. 2006]. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. doi: 10.17487/RFC4271. url: https://www.rfc-editor.org/info/rfc4271 [siehe S. 13, 14].

Schoch, Leon [Okt. 2022]. *API für Route Injection* [siehe S. 10, 11].