Fraunhofer FKIE

Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

28. VDE/ITG Fachtagung Mobilkommunikation, Mai 2024

Thorsten Horstmann

Implementation of OpenAPI Wireshark Dissectors to Validate SBI Messages of 5G Core Networks

Vorstellung

Unsere Forschungsgruppe: “Secure Mobile Communication”

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fraunhofer
FKIE

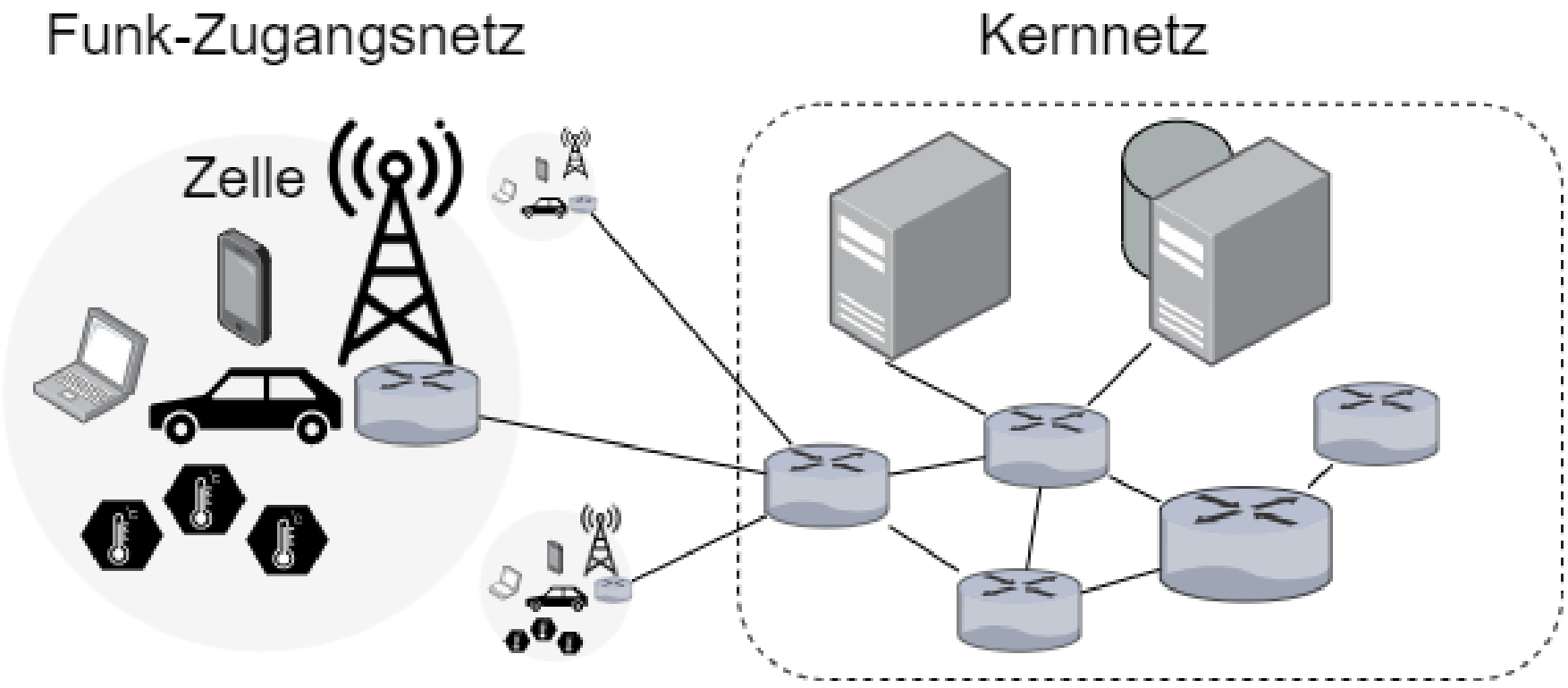
- **Fachbereich:** Informatik
- **Lehrstuhl:** Embedded Systems und Netze

- **Abteilung:** Cyber Analysis & Defense
- **Gruppe:** Secure Mobile Communication

Wir betreiben angewandte Forschung, um die Sicherheit und Skalierbarkeit von drahtlosen Kommunikationsnetzwerken zu analysieren und zu verbessern.

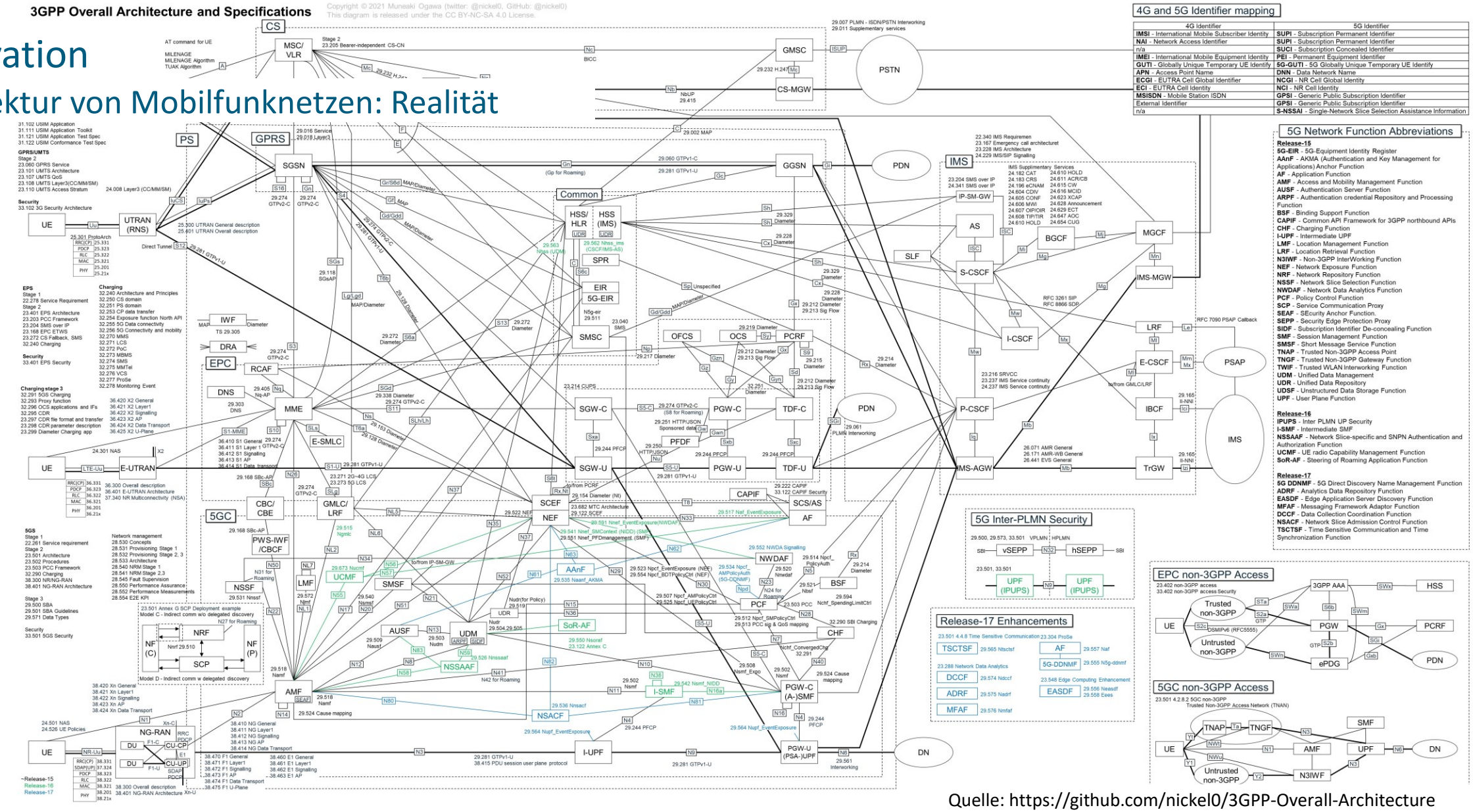
Motivation

Architektur von Mobilfunknetzen: Sehr stark vereinfacht



Motivation

Architektur von Mobilfunknetzen: Realität



Quelle: https://github.com/nickel0/3GPP-Overall-Architecture

Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

Fraunhofer FKIE

Motivation

Analyse der Nachrichten in einem 5G Kernnetzwerk

- Durch Aufteilung des 5G Kernnetzes in „Network Functions“ (NFs)
  - Viele verschiedene Nachrichten
  - Komplexe Sequenzen von Paketen zwischen den NFs, selbst für einfache Prozeduren
- Unterstützung in Wireshark
  - Gut für binär-codierte Nachrichten, dank ASN.1
  - Nicht vorhanden für HTTP basierte Pakete des „Service-based-Interface“ (SBI)
- Dadurch schwierige Fehlersuche
  - Keine Filter für interessante Felder innerhalb der SBI Pakete
  - Keine Gruppierung von HTTP/2 Request – Response Paaren

- Fehlende Validierung der Nachrichten auf Konformität
  - Entspricht der Aufbau der Nachrichten und der Felder den Spezifikationen?
  - Wichtig für Interoperabilität von NFs verschiedener Hersteller

OpenAPI
Grundlagen

- Standard für die Definition von RESTful APIs
  - sprachunabhängige Beschreibung der
    - API-Endpunkte
    - Request / Response Formate
    - Parameter
    - Authentifizierungsmethoden
  - Maschinenlesbar (JSON / YAML)
- Beschreibt „nur“ Requests-Response Paare, keine Ablaufsbeschreibung
- Verwendung:
  - Automatische Generierung von Client-Code
  - API Testing
  - Mocking / Stubs
  - Validierung

Wird in der 3GPP 5G Spezifikation für die Beschreibung des SBI verwendet

OpenAPI

Beispiel-Dokument

```yaml
openapi: 3.0.0
info:
  title: Sample API
  description: Example API to describe the idea.
  version: 0.1.9

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

- Metainformationen
  - Version, Titel, Beschreibung

- `paths` definiert einzelne Endpunkte der API
- Request-Infos
  - Indexiert anhand der Request-Pfade
  - Aufgesplittet je nach HTTP-Methode und Content-Type
  - Beschreibung des JSON Inhaltes

OpenAPI

Beispiel-Dokument

```yaml
paths:
  /users/{userId}:
    get:
      summary: Returns a user by ID.
      parameters:
        - name: userId
          in: path
          required: true
          description: The ID of the user to return.
          schema:
            type: integer
            format: int64
            minimum: 1
      responses:
        '200':
          …
        '400':
          description: The specified user ID is invalid (not a number).
        '404':
          description: A user with the specified ID was not found.
```

- Request-Infos
  - Beschreibung von Parametern
    - Hier am Beispiel in der URL codiert
- Response-Info
  - In den jeweiligen Request-Infos eingeordnet
  - Aufgesplittet anhand von Status-Code und Content-Type

OpenAPI

Beispiel-Dokument

```yaml
components:
  schemas:
    User:
      type: object
      properties:
        id:
          type: integer
          example: 4
        name:
          type: string
          example: Arthur Dent
      required:
        - id
        - name
```

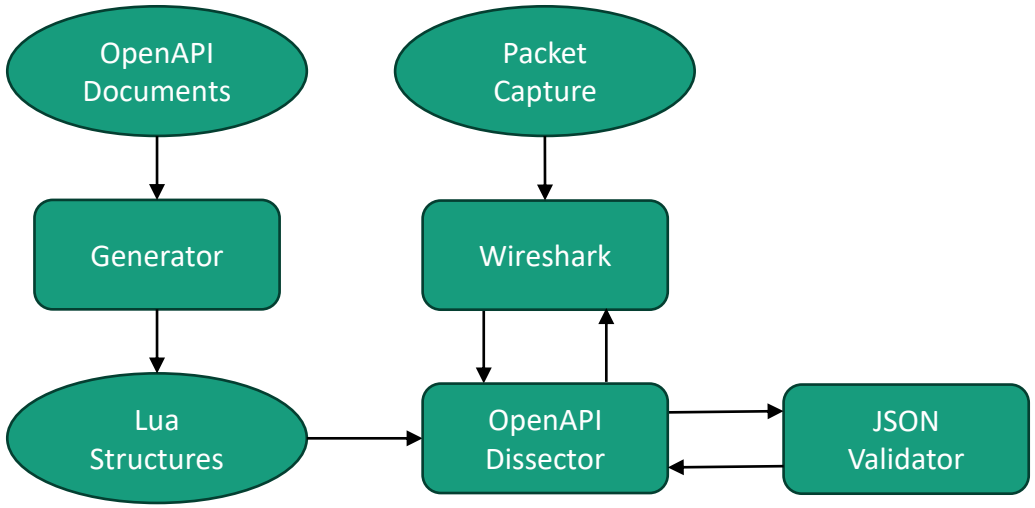
```yaml
paths:
  /users:
    post:
      summary: Creates a new user.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
      responses:
        '201':
          description: Created
```

- `components/schemas` erlaubt Definition wiederverwendbare Beschreibungen
- Starke Verwendung in den 5G Spezifikationen
  - Mehrstufige Referenzen
  - Dokumentübergreifend
- Weitere komplexe Beschreibungsmerkmale
  - Reguläre Ausdrücke
  - `anyOf`/`allOf`/`oneOf`/`not` Schlüsselwörter
  - Callbacks
  - …

Wireshark OpenAPI Dissector

Aufbau

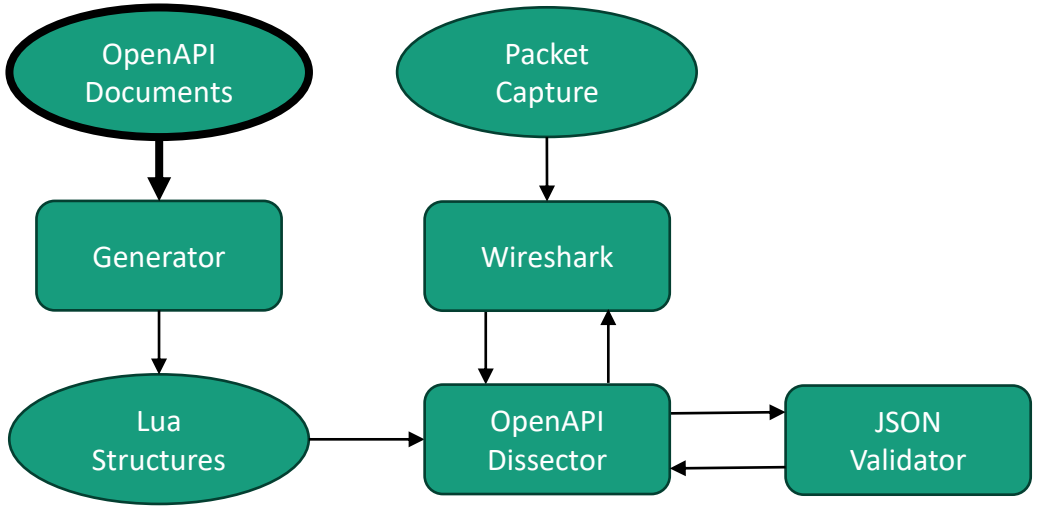
- Wireshark
  - DAS Werkzeug zur Netzwerkanalyse
  - Keine Unterstützung für OpenAPI basierte Protokolle
  - Erweiterbar durch Dissector-Plugins (Binary oder Lua)
- OpenAPI Dissector
  - Lua-Skript als Post-Dissector
    - Portabilität
  - Pre-Processing mit Python (Vorbereitung der OpenAPI-Dokumente)
  - Kompatibel mit 3GPP 5G OpenAPI-Dokumenten



Wireshark OpenAPI Dissector

3GPP 5G OpenAPI Dokumente

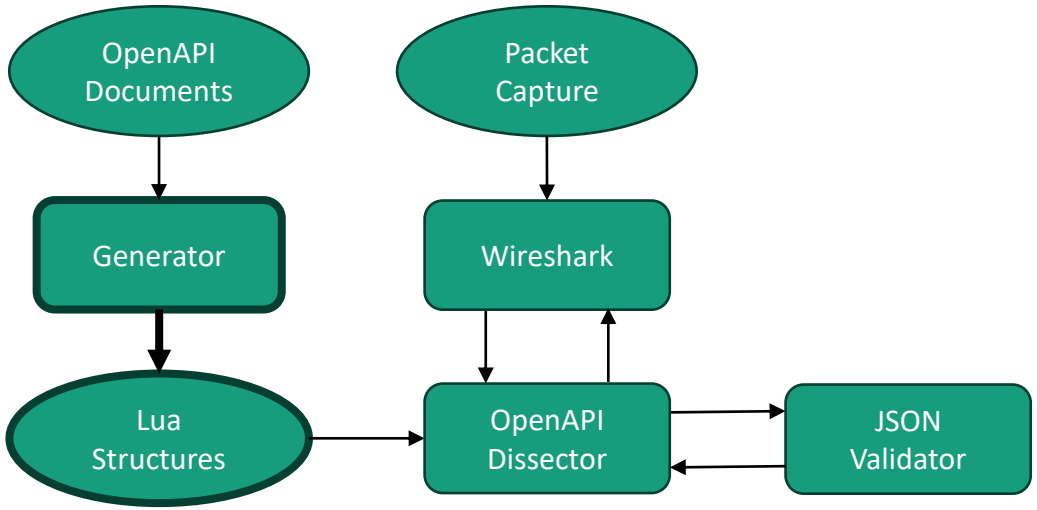
- In den ersten Releases der 5G Spezifikationen
  - Nur eingebettet in PDFs
  - Fehlerhaft
- Mittlerweile separat gepflegt frei verfügbar
  - https://forge.3gpp.org/rep/all/5G_APIs specs/



Wireshark OpenAPI Dissector

Generator

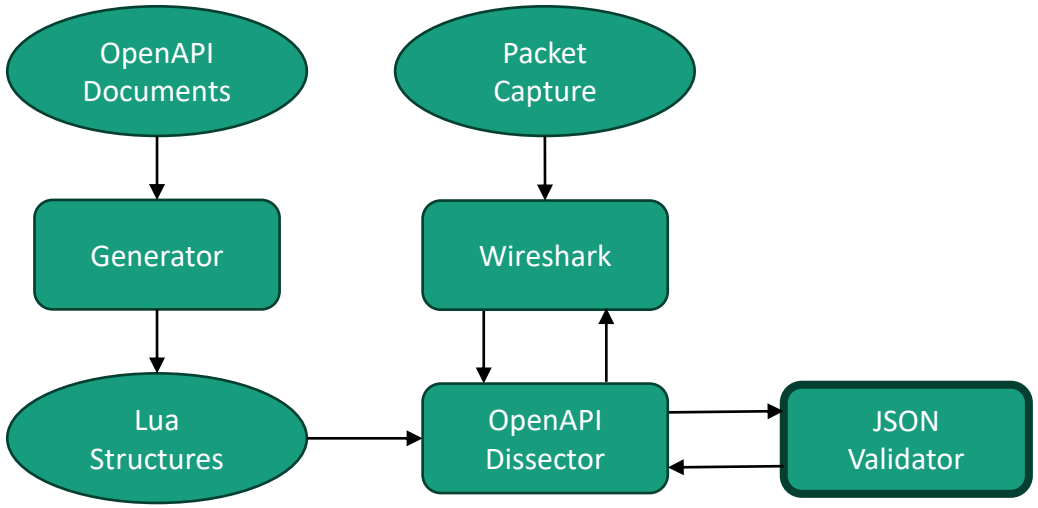
- Aufbereitung der OpenAPI-Dokumente für schnelleren Zugriff
  - Generiert Lua-Strukturen, welche vom eigentlichen Dissector verwendet werden
  - Auflösen aller verwendeten OpenAPI `components` Referenzen
- Konvertierung von Parametern in Pfaden zu regulären Ausdrücken



Wireshark OpenAPI Dissector

JSON Validator

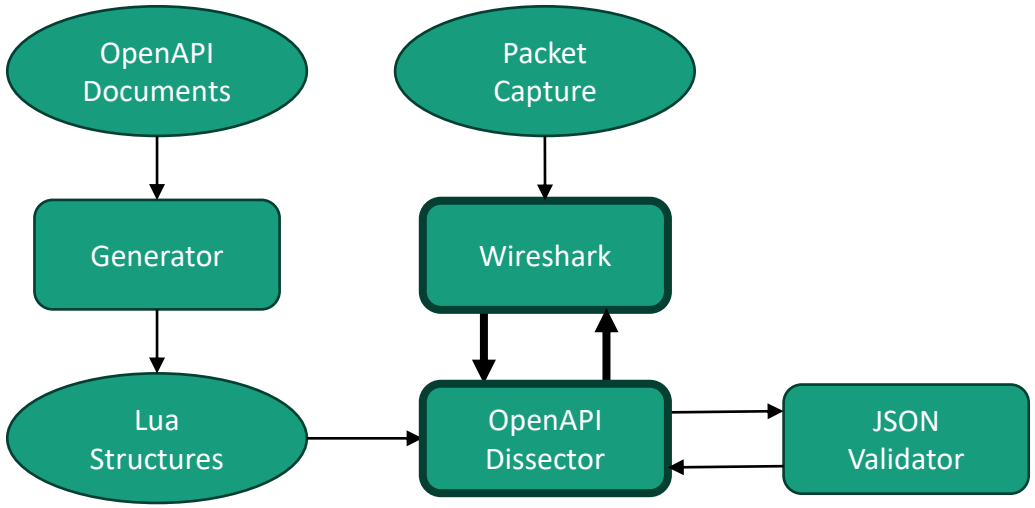
- Abgleich der HTTP/2 Inhalte gegen die Spezifikation(en)
  - JSON Parsing durch externe Bibliothek
  - Validierungslogik selbst entwickelt
- Generierung von Fehler-Informationen bei Abweichungen



Wireshark OpenAPI Dissector

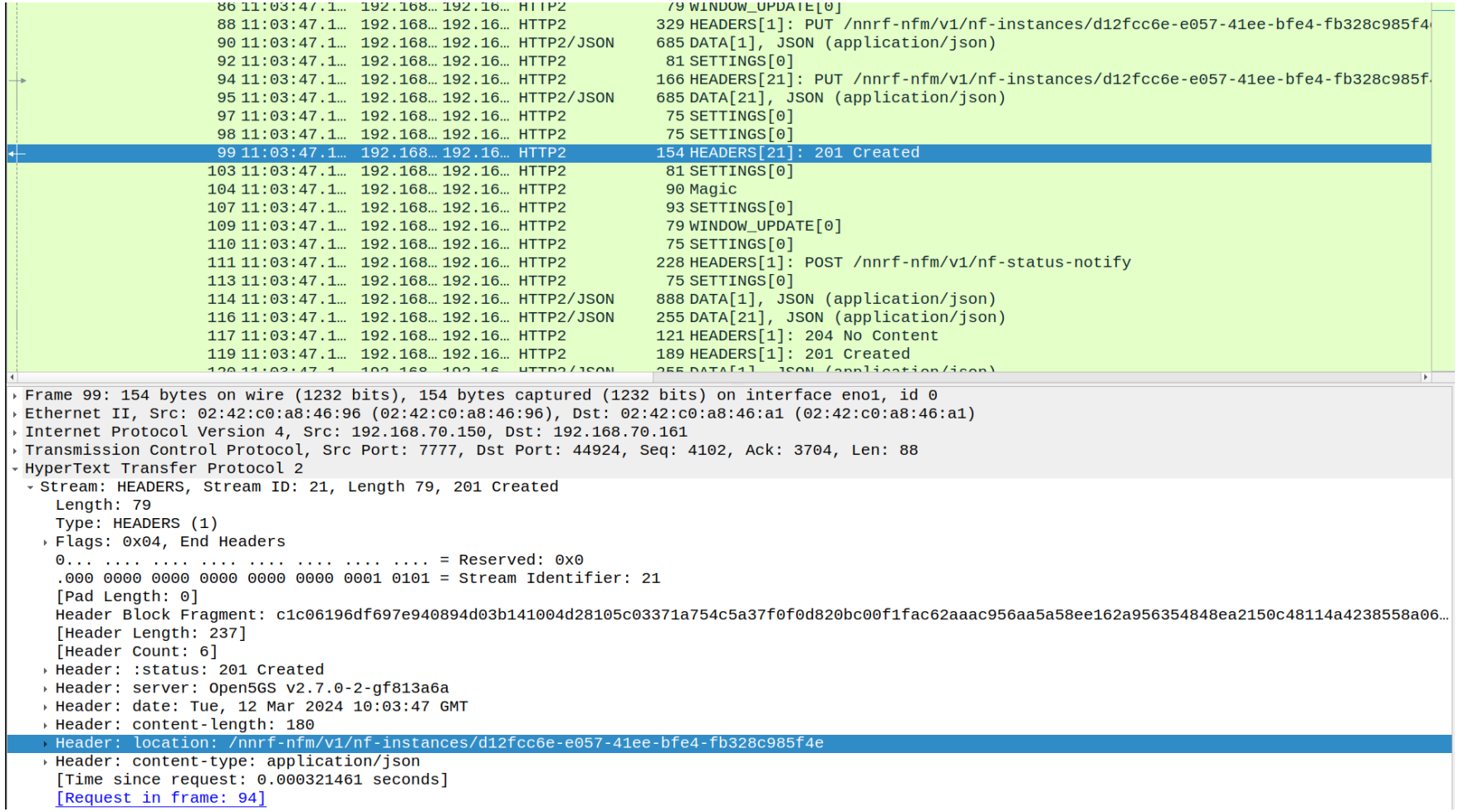
Haupt-Plugin

- Auslesen der HTTP/2 Segmente aus dem Wireshark Dissection-Tree
- Korrektur fehlender Header (Komprimierung in HTTP/2)
- “Erraten” von Content-Type Headern bei fehlerhafter oder gar keiner Angabe
  - Häufig `application/json` statt der in den Spezifikationen vorgegebenen Typen
- Darstellung der geparsten Informationen und des Validierungs-Zustands



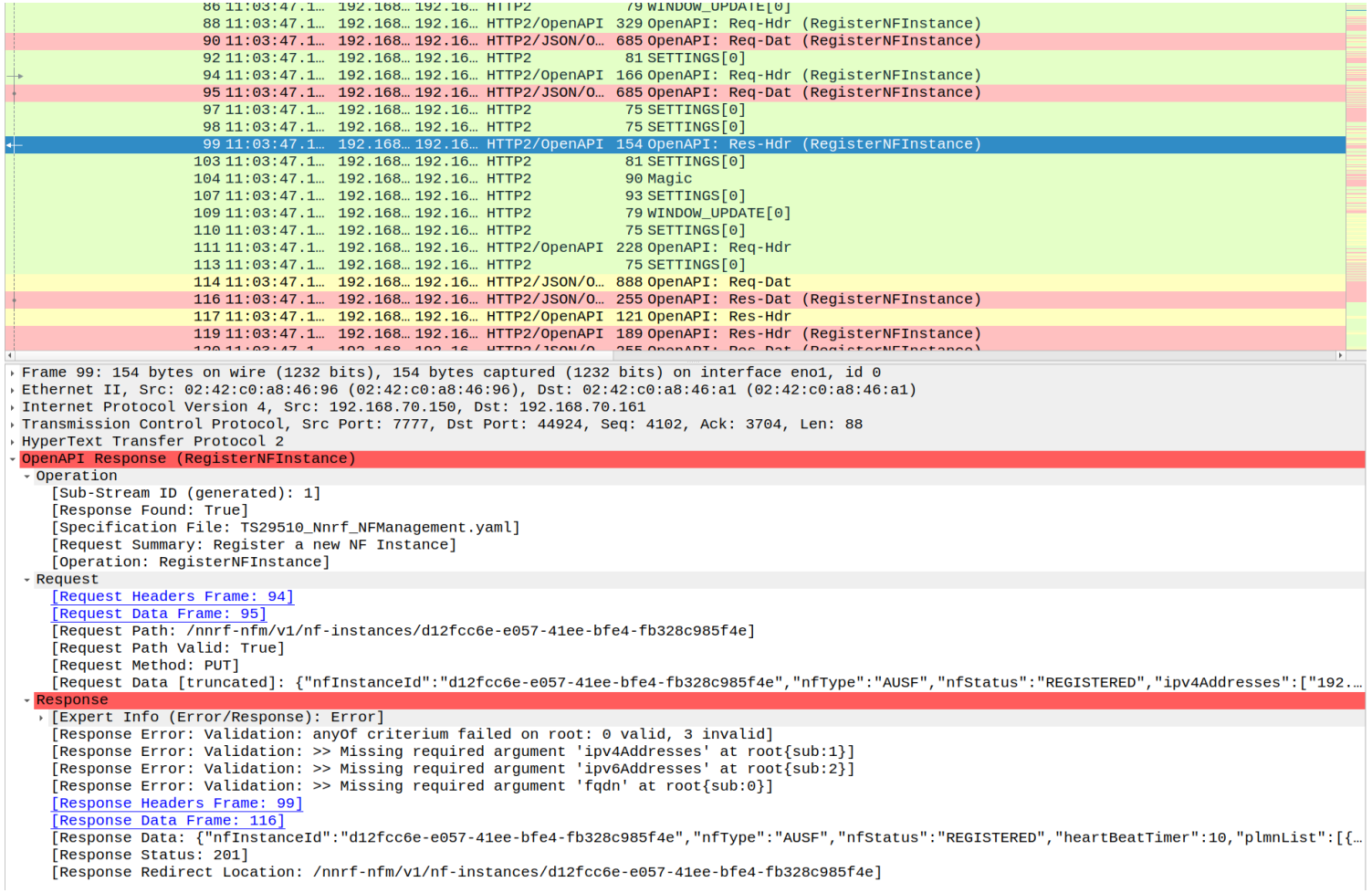
Analyse eines 5G Mitschnitts

Ohne OpenAPI Dissector



Analyse eines 5G Mitschnitts

Mit OpenAPI Dissector

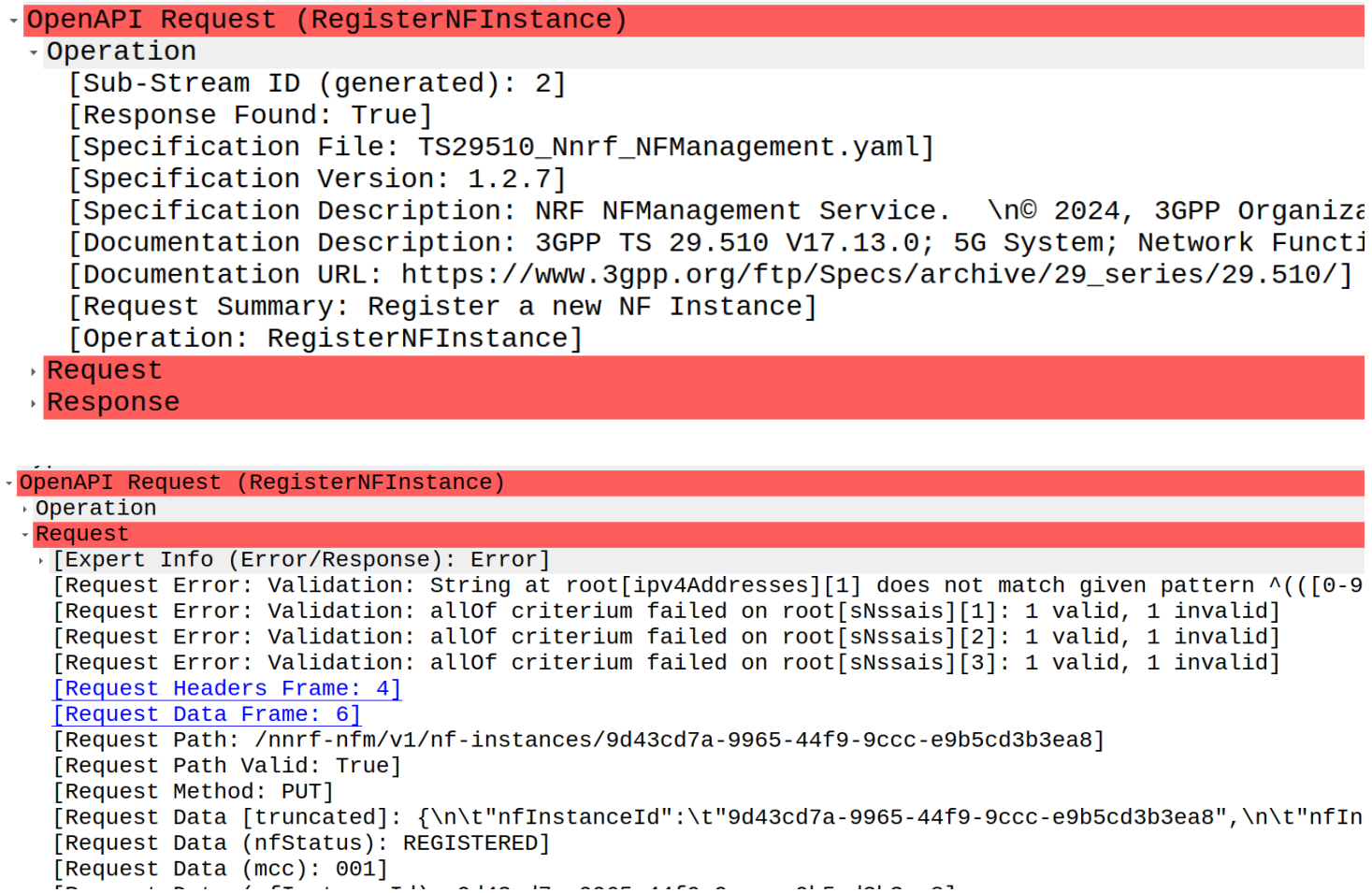


```
 86 11:03:47.1… 192.168… 192.16… HTTP2           79 WINDOW_UPDATE[0]
 88 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  329 OpenAPI: Req-Hdr (RegisterNFInstance)
 90 11:03:47.1… 192.168… 192.16… HTTP2/JSON/O…  685 OpenAPI: Req-Dat (RegisterNFInstance)
 92 11:03:47.1… 192.168… 192.16… HTTP2           81 SETTINGS[0]
 94 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  166 OpenAPI: Req-Hdr (RegisterNFInstance)
 95 11:03:47.1… 192.168… 192.16… HTTP2/JSON/O…  685 OpenAPI: Req-Dat (RegisterNFInstance)
 97 11:03:47.1… 192.168… 192.16… HTTP2           75 SETTINGS[0]
 98 11:03:47.1… 192.168… 192.16… HTTP2           75 SETTINGS[0]
 99 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  154 OpenAPI: Res-Hdr (RegisterNFInstance)
103 11:03:47.1… 192.168… 192.16… HTTP2           81 SETTINGS[0]
104 11:03:47.1… 192.168… 192.16… HTTP2           90 Magic
107 11:03:47.1… 192.168… 192.16… HTTP2           93 SETTINGS[0]
109 11:03:47.1… 192.168… 192.16… HTTP2           79 WINDOW_UPDATE[0]
110 11:03:47.1… 192.168… 192.16… HTTP2           75 SETTINGS[0]
111 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  228 OpenAPI: Req-Hdr
113 11:03:47.1… 192.168… 192.16… HTTP2           75 SETTINGS[0]
114 11:03:47.1… 192.168… 192.16… HTTP2/JSON/O…  888 OpenAPI: Req-Dat
116 11:03:47.1… 192.168… 192.16… HTTP2/JSON/O…  255 OpenAPI: Res-Dat (RegisterNFInstance)
117 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  121 OpenAPI: Res-Hdr
119 11:03:47.1… 192.168… 192.16… HTTP2/OpenAPI  189 OpenAPI: Res-Hdr (RegisterNFInstance)

▸ Frame 99: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface eno1, id 0
▸ Ethernet II, Src: 02:42:c0:a8:46:96 (02:42:c0:a8:46:96), Dst: 02:42:c0:a8:46:a1 (02:42:c0:a8:46:a1)
▸ Internet Protocol Version 4, Src: 192.168.70.150, Dst: 192.168.70.161
▸ Transmission Control Protocol, Src Port: 7777, Dst Port: 44924, Seq: 4102, Ack: 3704, Len: 88
▸ HyperText Transfer Protocol 2
▾ OpenAPI Response (RegisterNFInstance)
  ▾ Operation
      [Sub-Stream ID (generated): 1]
      [Response Found: True]
      [Specification File: TS29510_Nnrf_NFManagement.yaml]
      [Request Summary: Register a new NF Instance]
      [Operation: RegisterNFInstance]
  ▾ Request
      [Request Headers Frame: 94]
      [Request Data Frame: 95]
      [Request Path: /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4e]
      [Request Path Valid: True]
      [Request Method: PUT]
      [Request Data [truncated]: {"nfInstanceId":"d12fcc6e-e057-41ee-bfe4-fb328c985f4e","nfType":"AUSF","nfStatus":"REGISTERED","ipv4Addresses":["192.…
  ▾ Response
    ▸ [Expert Info (Error/Response): Error]
      [Response Error: Validation: anyOf criterium failed on root: 0 valid, 3 invalid]
      [Response Error: Validation: >> Missing required argument 'ipv4Addresses' at root{sub:1}]
      [Response Error: Validation: >> Missing required argument 'ipv6Addresses' at root{sub:2}]
      [Response Error: Validation: >> Missing required argument 'fqdn' at root{sub:0}]
      [Response Headers Frame: 99]
      [Response Data Frame: 116]
      [Response Data: {"nfInstanceId":"d12fcc6e-e057-41ee-bfe4-fb328c985f4e","nfType":"AUSF","nfStatus":"REGISTERED","heartBeatTimer":10,"plmnList":[{…
      [Response Status: 201]
      [Response Redirect Location: /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4e]
```

Analyse eines 5G Mitschnitts

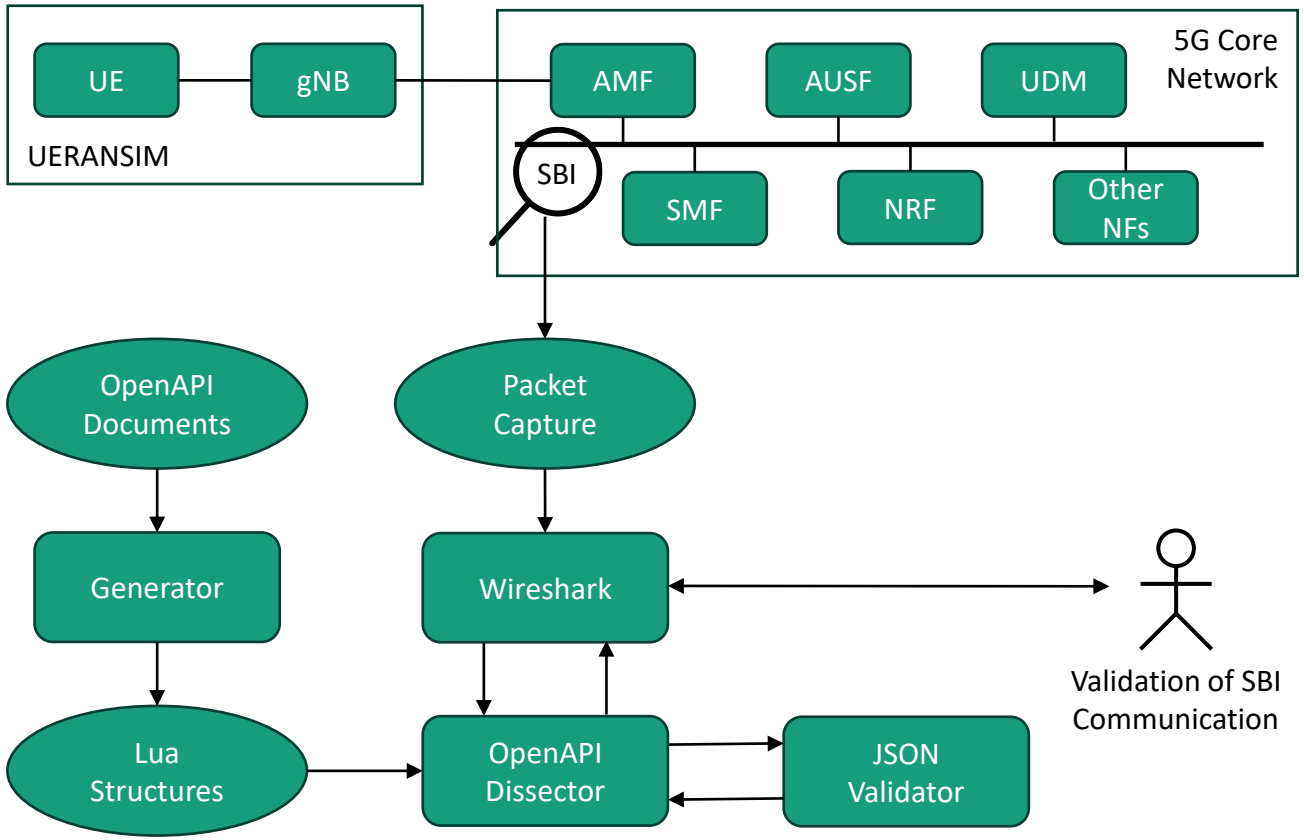
Mit OpenAPI Dissector



Testen des Dissectors

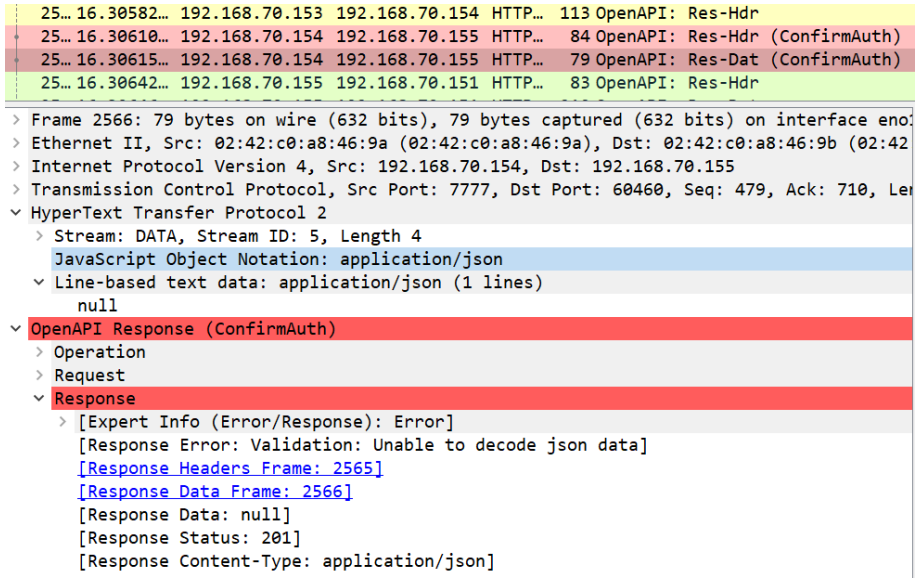
Evaluierung von open-source Kernnetzwerken

- Evaluierung von drei open-source Projekten:
  - Open5GS (v2.7.0)
  - free5GC (v3.3.0)
  - OpenAirInterface (v2.0.1)
- Einrichtung von Testnetzwerken mit
  - der jeweiligen 5G Core Implementierung
  - UERANSIM
    - Emuliert das Verhalten von gNB und UE
    - Kein Funk, dadurch hohe Reproduzierbarkeit
- Testszenario:
  - Initialisierung des 5G-Netzwerks
  - UE-Registrierung mit PDU-Session-Einrichtung
  - Deregistrierung des simulierten UEs

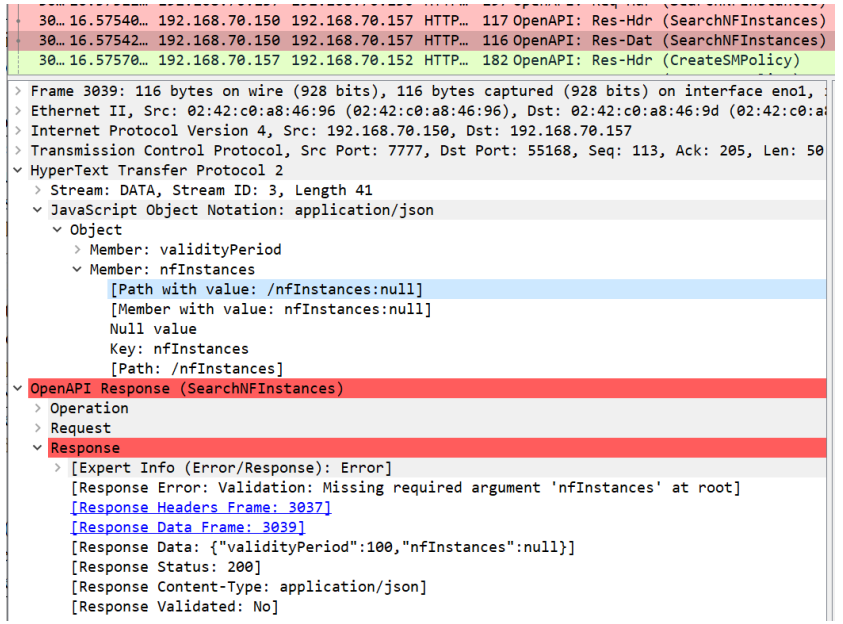


Ausgewählte Ergebnisse
„null“ in free5GC

- Anstelle einer gültigen `AuthEvent`-Nachricht ist die Antwort auf `confirmAuth`-Nachrichten einfach der Wert `null`
- Erwarteter Location-Header zu der erstellten Ressource fehlt ebenfalls



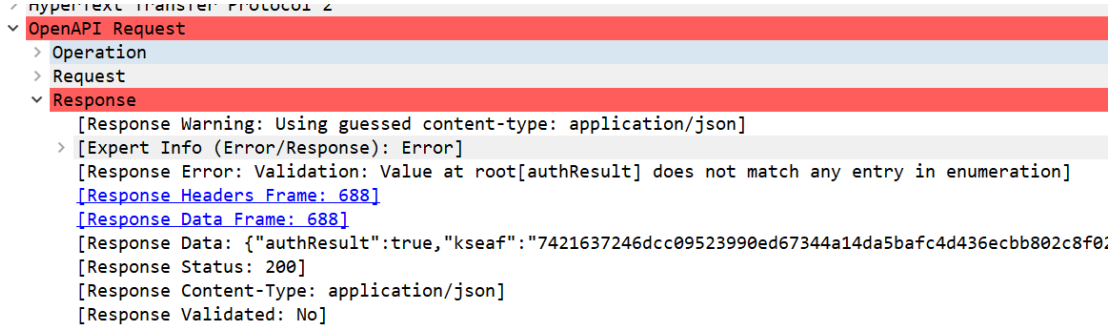
- Anstelle einer leeren `nfInstances` Liste enthält die Antwort auf eine `SearchNFInstances` einfach den Wert `null`



Ausgewählte Ergebnisse

Tüte Gemischte Fehler

- Alle Implementierungen
  - Falsche API Version in URL bei Zugriff auf UDR Endpunkte
  - Falsche `content-type` Header
- Open5GS
  - Leere `nfServiceList` während `RegisterNFInstance`
  - Falsches Format der NRF `subscriptionId`
  - Antwort auf `RegisterNFInstance` fehlen Adressinformationen
- free5GC
  - Negatives `ageOfLocationInformation` in `UpdateSmContext`
- OpenAirInterface
  - Ungültige `SupiRange` Beschreibung
  - Ungültige Slice-Differentiator Codierung
  - Antwort auf `5g-aka-confirmation` enthält `Boolean` anstatt `Enumeration`



Ausgewählte Ergebnisse

Nutzung von oneOf in OpenAirInterface

- Gültigkeit mehrerer verschiedener Datenstrukturen auf einem Endpunkt
- Auswahl durch Discriminator-Attribut
  - z.B. für verschiedene Authentifizierungsmechanismen
- Anwendung auf enthaltendes Element, nicht auf neu erzeugtes Unter-Element
- Häufig Erzeugung fehlerhafter untergeordneter Objekte
  - Limitierung in der OpenAPI-Generator Bibliothek
  - Hunderte offene GitHub-Issues dazu

- Master Thesis zeigte ähnlichen Fehler in Ericssons Core Implementierung
- Weiteres betroffenes kommerzielles Kernnetzwerk

OpenAPI Spezifikation

```
AuthenticationVector:
  oneOf:
    - $ref: '#/components/schemas/Av5GHeAka'
    - $ref: '#/components/schemas/AvEapAkaPrime'
  discriminator:
    propertyName: avType
    mapping:
      5G_HE_AKA: '#/components/schemas/Av5GHeAka'
      EAP_AKA_PRIME: '#/components/schemas/AvEapAkaPrime'
```

Korrekte Datenstruktur

```
{
  "authType": "5G_AKA",
  "authenticationVector": {
    "avType": "5G_HE_AKA",
    "rand": "123",
    "xresStar": "456",
    "autn": "789",
    "kausf": "abc"
  },
  "supi": "imsi-001010[...]"
}
```

Verwendete Datenstruktur

```
{
  "authType": "5G_AKA",
  "authenticationVector": {
    "Av5GHeAka": {
      "avType": "5G_HE_AKA",
      "rand": "123",
      "xresStar": "456",
      "autn": "789",
      "kausf": "abc"
    }
  },
  "supi": "imsi-001010[...]"
}
```

Limitierungen und Ausblick

Abgeschlossene Punkte
- Auswahl des 3GPP Releases in Wireshark
- Definition eigener Filter
  - `openapi_fields["smfId"] = {"all[nfInstanceId]", "all[smfId]"}`

Offene Punkte
- Aktuell nur HTTP/2 Unterstützung
  - Analyse von 5G war primärer Anwendungsfall
- Unvollständige Unterstützung für `anyOf/allOf/oneOf/not` Schlüsselwörter
- Keine Unterstützung für vordefinierte Datentypen (z.B. email, UUID, ...)

Ausblick
- Gefundene Fehler den Projekten melden
- Integration in die Testumgebung der Deutschen Telekom
  - Nicht-grafische Nutzung des Dissectors
  - Maschinenlesbare Ausgabe der Analyse
  - „Validation-as-a-Service“
- Dissector als Ausgangspunkt für die Analyse von Network-Function übergreifende Prozeduren

Einladung den Dissector auszuprobieren
- Wir freuen uns über Feedback
- Validierung von weiteren Implementierungen

https://github.com/telekom/OpenAPI-Dissector

Vielen Dank für Ihre Aufmerksamkeit!
Fragen? Ideen? Anregungen?

Kontakt

Thorsten Horstmann

Fraunhofer FKIE
Cyber Analysis & Defense
thorsten.horstmann@fkie.fraunhofer.de

Fraunhofer-Institut für Kommunikation,
Informationsverarbeitung und Ergonomie FKIE
Fraunhoferstr. 20 | 53343 Wachtberg

www.fkie.fraunhofer.de

Hochschule Bonn-Rhein-Sieg
Institut für Sicherheitsforschung (ISF)
thorsten.horstmann@h-brs.de

Hochschule Bonn-Rhein-Sieg
Grantham-Allee 20 | 53757 Sankt Augustin

www.h-brs.de