

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

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



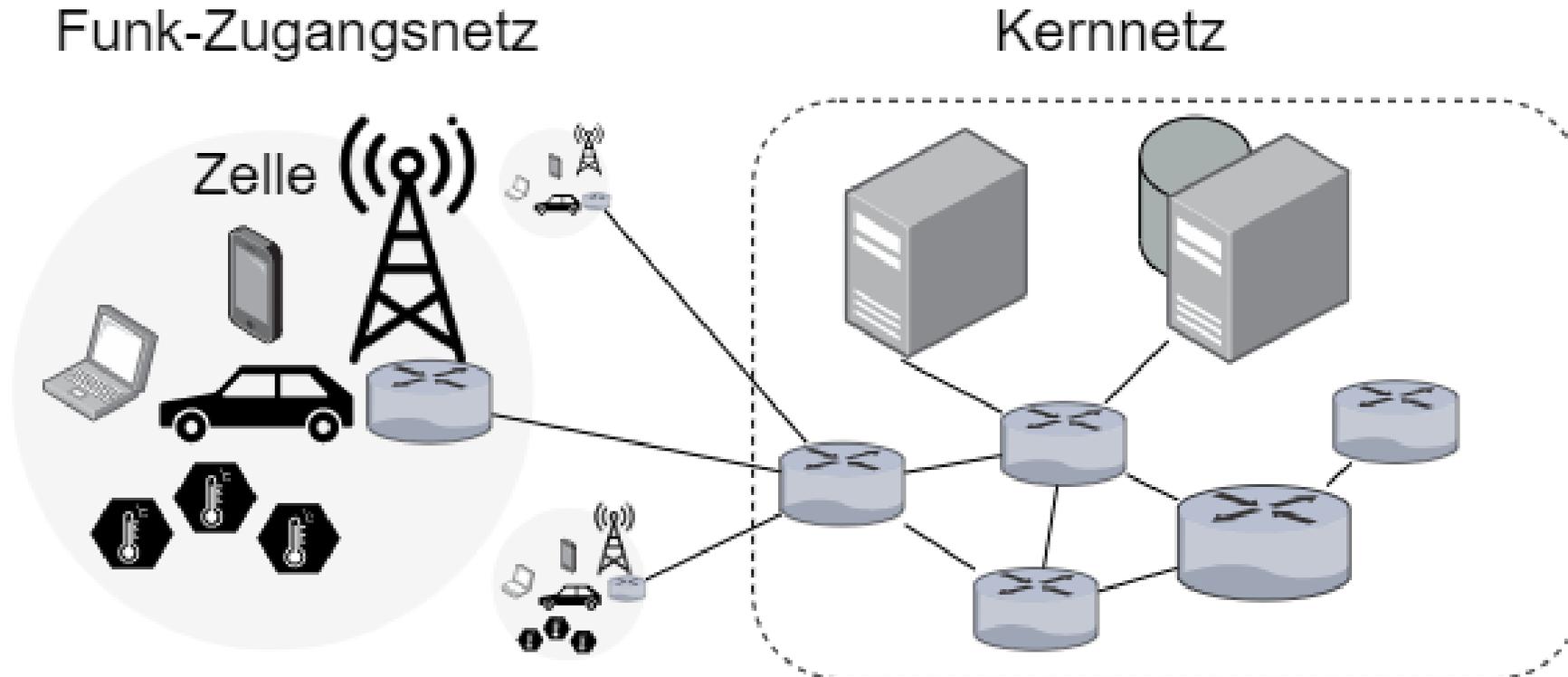
Fraunhofer
FKIE

- **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

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 Ablaufbeschreibung
- 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

```
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

```
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

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

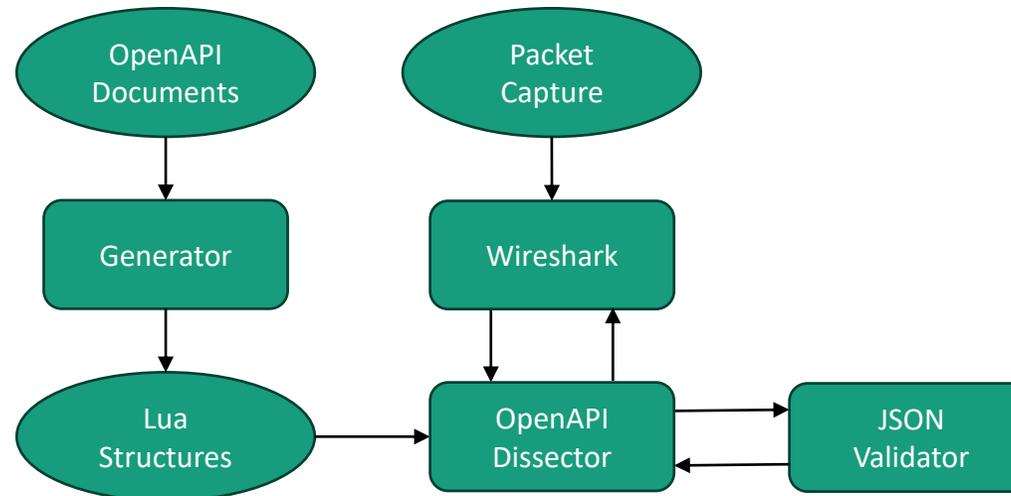
  paths:
    /users:
      post:
        summary: Creates a new user.
        requestBody:
          required: true
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
        responses:
          '201':
            description: Created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

- 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

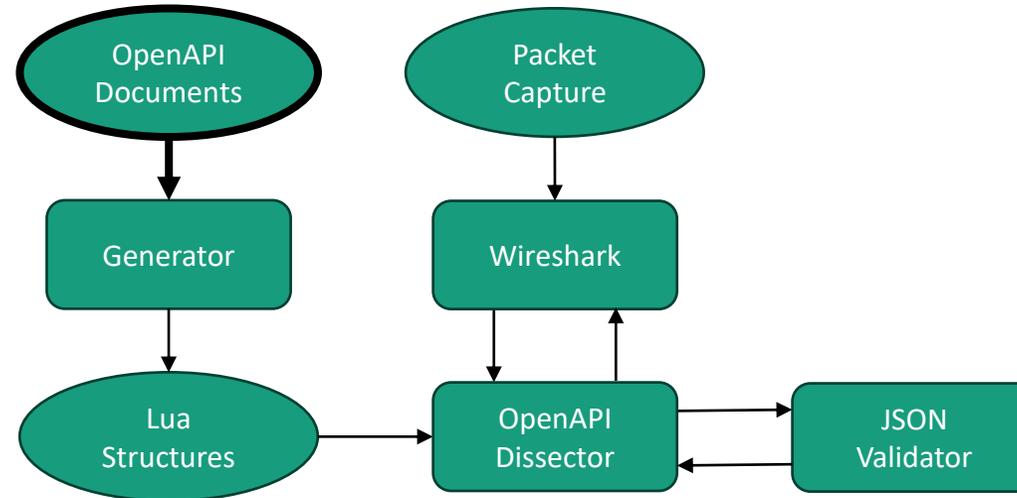
- Wireshark
 - DAS Werkzeug zur Netzwerkanalyse
 - Keine Unterstützung für OpenAPI basierte Protokolle
 - Erweiterbar durch Dissector-Plugins (Binary oder Lua)
- OpenAPI Dissector
 - Lua-Skript als Post-Dissector
 - Portabilität
 - Pre-Processing mit Python (Vorbereitung der OpenAPI-Dokumente)
 - Kompatibel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



Wireshark OpenAPI Dissector

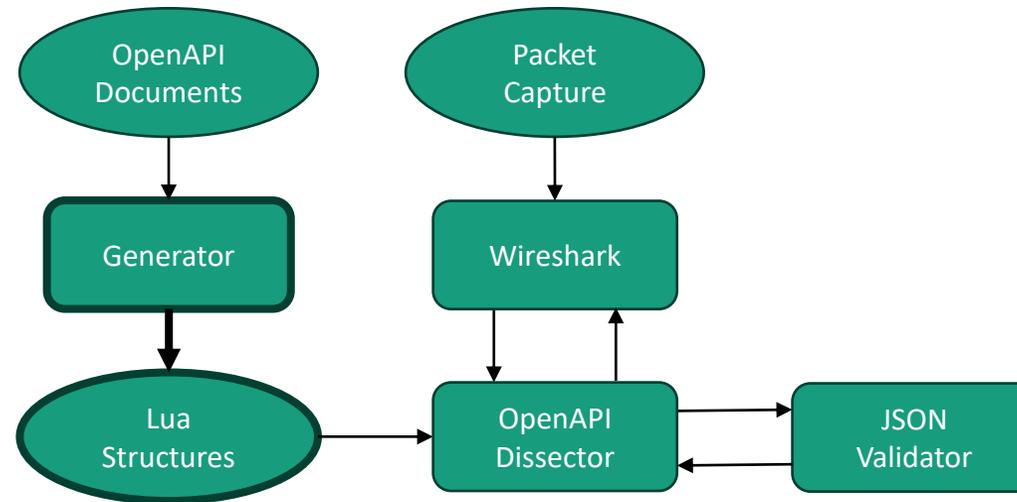
3GPP 5G OpenAPI Dokumente

- 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

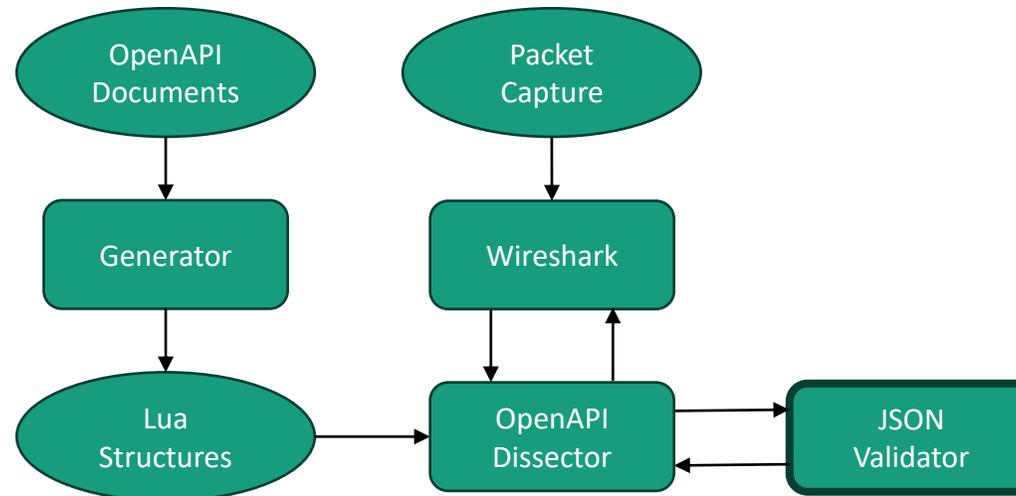
- Aufbereitung der OpenAPI-Dokumente für schnelleren Zugriff
 - Generiert Lua-Strukturen, welche vom eigentlichen Dissector verwendet werden
 - Auflösen aller verwendeten OpenAPI components Referenzen
- Konvertierung von Parametern in Pfaden zu regulären Ausdrücken



Wireshark OpenAPI Dissector

JSON Validator

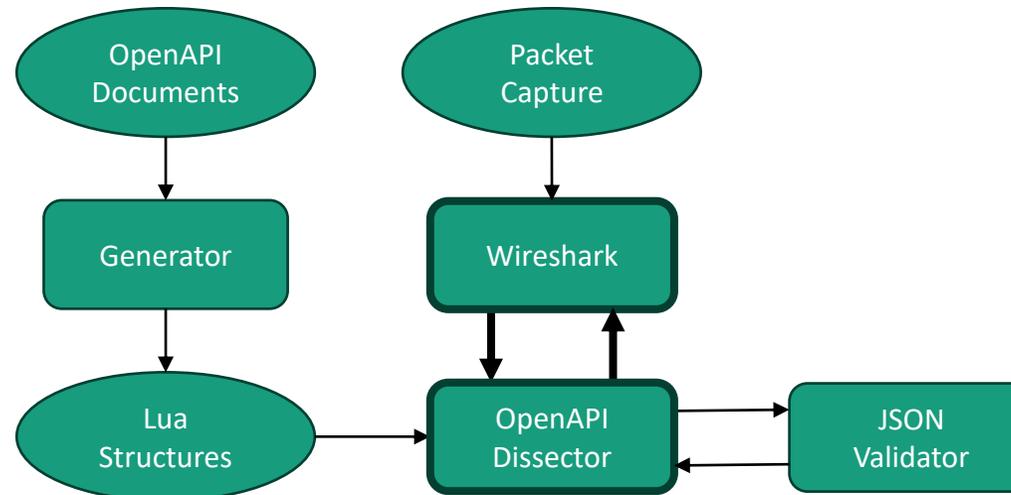
- Abgleich der HTTP/2 Inhalte gegen die Spezifikation(en)
 - JSON Parsing durch externe Bibliothek
 - Validierungslogik selbst entwickelt
- Generierung von Fehler-Informationen bei Abweichungen



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

```
86 11:03:47.1... 192.168... 192.16... HTTP2 79 WINDOW_UPDATE[0]
88 11:03:47.1... 192.168... 192.16... HTTP2 329 HEADERS[1]: PUT /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4
90 11:03:47.1... 192.168... 192.16... HTTP2/JSON 685 DATA[1], JSON (application/json)
92 11:03:47.1... 192.168... 192.16... HTTP2 81 SETTINGS[0]
94 11:03:47.1... 192.168... 192.16... HTTP2 166 HEADERS[21]: PUT /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4
95 11:03:47.1... 192.168... 192.16... HTTP2/JSON 685 DATA[21], JSON (application/json)
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 154 HEADERS[21]: 201 Created
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 228 HEADERS[1]: POST /nnrf-nfm/v1/nf-status-notify
113 11:03:47.1... 192.168... 192.16... HTTP2 75 SETTINGS[0]
114 11:03:47.1... 192.168... 192.16... HTTP2/JSON 888 DATA[1], JSON (application/json)
116 11:03:47.1... 192.168... 192.16... HTTP2/JSON 255 DATA[21], JSON (application/json)
117 11:03:47.1... 192.168... 192.16... HTTP2 121 HEADERS[1]: 204 No Content
119 11:03:47.1... 192.168... 192.16... HTTP2 189 HEADERS[1]: 201 Created
120 11:03:47.1... 192.168... 192.16... HTTP2/JSON 255 DATA[1], JSON (application/json)
```

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
- Stream: HEADERS, Stream ID: 21, Length 79, 201 Created
Length: 79
Type: HEADERS (1)
Flags: 0x04, End Headers
0... .. = Reserved: 0x0
.000 0000 0000 0000 0000 0000 0001 0101 = Stream Identifier: 21
[Pad Length: 0]
Header Block Fragment: c1c06196df697e940894d03b141004d28105c03371a754c5a37f0f0d820bc00f1fac62aaac956aa5a58ee162a956354848ea2150c48114a4238558a06...
[Header Length: 237]
[Header Count: 6]
Header: :status: 201 Created
Header: server: Open5GS v2.7.0-2-gf813a6a
Header: date: Tue, 12 Mar 2024 10:03:47 GMT
Header: content-length: 180
Header: location: /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4e
Header: content-type: application/json
[Time since request: 0.000321461 seconds]
[Request in frame: 94]

Analyse eines 5G Mitschnitts

Mit OpenAPI Dissector

The image displays a Wireshark packet capture of an OpenAPI response. The top section shows a list of packets with columns for Time, Source, Destination, Protocol, and Length. Packet 99 is highlighted in blue, indicating it is the selected packet. Below the packet list, the details pane for packet 99 is expanded, showing the following structure:

- 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":["192.168.70.150"]}]
 - 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":[{"mcc":310,"mnc":150,"plmn":310150100}]}]
 - [Response Status: 201]
 - [Response Redirect Location: /nnrf-nfm/v1/nf-instances/d12fcc6e-e057-41ee-bfe4-fb328c985f4e]

Analyse eines 5G Mitschnitts

Mit OpenAPI Dissector

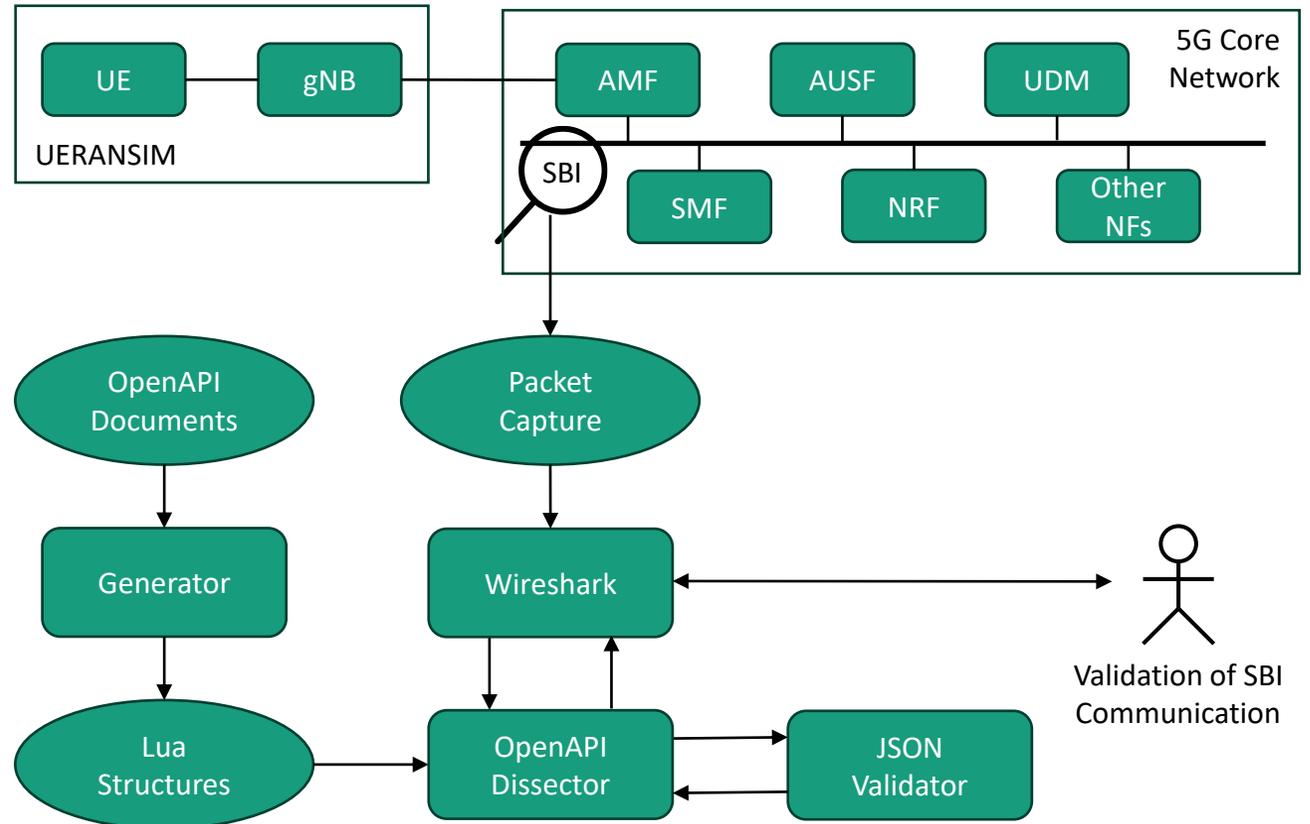
```
- OpenAPI Request (RegisterNFInstance)
- Operation
  [Sub-Stream ID (generated): 2]
  [Response Found: True]
  [Specification File: TS29510_Nnrf_NFManagement.yaml]
  [Specification Version: 1.2.7]
  [Specification Description: NRF NFManagement Service. \n© 2024, 3GPP Organiza
  [Documentation Description: 3GPP TS 29.510 V17.13.0; 5G System; Network Functi
  [Documentation URL: https://www.3gpp.org/ftp/Specs/archive/29_series/29.510/]
  [Request Summary: Register a new NF Instance]
  [Operation: RegisterNFInstance]
  ▸ Request
  ▸ Response

- OpenAPI Request (RegisterNFInstance)
  ▸ Operation
  ▸ Request
    [Expert Info (Error/Response): Error]
    [Request Error: Validation: String at root[ipv4Addresses][1] does not match given pattern ^(([0-9
    [Request Error: Validation: allof criterium failed on root[sNssais][1]: 1 valid, 1 invalid]
    [Request Error: Validation: allof criterium failed on root[sNssais][2]: 1 valid, 1 invalid]
    [Request Error: Validation: allof criterium failed on root[sNssais][3]: 1 valid, 1 invalid]
    [Request Headers Frame: 4]
    [Request Data Frame: 6]
    [Request Path: /nnrf-nfm/v1/nf-instances/9d43cd7a-9965-44f9-9ccc-e9b5cd3b3ea8]
    [Request Path Valid: True]
    [Request Method: PUT]
    [Request Data [truncated]: {\n\t"nfInstanceId":\t"9d43cd7a-9965-44f9-9ccc-e9b5cd3b3ea8",\n\t"nfIn
    [Request Data (nfStatus): REGISTERED]
    [Request Data (mcc): 001]
```

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

```
25... 16.30582... 192.168.70.153 192.168.70.154 HTTP... 113 OpenAPI: Res-Hdr
25... 16.30610... 192.168.70.154 192.168.70.155 HTTP... 84 OpenAPI: Res-Hdr (ConfirmAuth)
25... 16.30615... 192.168.70.154 192.168.70.155 HTTP... 79 OpenAPI: Res-Dat (ConfirmAuth)
25... 16.30642... 192.168.70.155 192.168.70.151 HTTP... 83 OpenAPI: Res-Hdr

> Frame 2566: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface eno1
> Ethernet II, Src: 02:42:c0:a8:46:9a (02:42:c0:a8:46:9a), Dst: 02:42:c0:a8:46:9b (02:42:c0:a8:46:9b)
> Internet Protocol Version 4, Src: 192.168.70.154, Dst: 192.168.70.155
> Transmission Control Protocol, Src Port: 7777, Dst Port: 60460, Seq: 479, Ack: 710, Len: 79
> HyperText Transfer Protocol 2
  > Stream: DATA, Stream ID: 5, Length 4
  JavaScript Object Notation: application/json
  > Line-based text data: application/json (1 lines)
    null
  > OpenAPI Response (ConfirmAuth)
    > Operation
    > Request
    > Response
      > [Expert Info (Error/Response): Error]
      [Response Error: Validation: Unable to decode json data]
      [Response Headers Frame: 2565]
      [Response Data Frame: 2566]
      [Response Data: null]
      [Response Status: 201]
      [Response Content-Type: application/json]
```

```
30... 16.57540... 192.168.70.150 192.168.70.157 HTTP... 117 OpenAPI: Res-Hdr (SearchNFInstances)
30... 16.57542... 192.168.70.150 192.168.70.157 HTTP... 116 OpenAPI: Res-Dat (SearchNFInstances)
30... 16.57570... 192.168.70.157 192.168.70.152 HTTP... 182 OpenAPI: Res-Hdr (CreateSMPolicy)

> Frame 3039: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface eno1
> Ethernet II, Src: 02:42:c0:a8:46:96 (02:42:c0:a8:46:96), Dst: 02:42:c0:a8:46:9d (02:42:c0:a8:46:9d)
> Internet Protocol Version 4, Src: 192.168.70.150, Dst: 192.168.70.157
> Transmission Control Protocol, Src Port: 7777, Dst Port: 55168, Seq: 113, Ack: 205, Len: 50
> HyperText Transfer Protocol 2
  > Stream: DATA, Stream ID: 3, Length 41
  JavaScript Object Notation: application/json
  > Object
    > Member: validityPeriod
    > Member: nfInstances
      [Path with value: /nfInstances:null]
      [Member with value: nfInstances:null]
      Null value
      Key: nfInstances
      [Path: /nfInstances]
  > OpenAPI Response (SearchNFInstances)
    > Operation
    > Request
    > Response
      > [Expert Info (Error/Response): Error]
      [Response Error: Validation: Missing required argument 'nfInstances' at root]
      [Response Headers Frame: 3037]
      [Response Data Frame: 3039]
      [Response Data: {"validityPeriod":100,"nfInstances":null}]
      [Response Status: 200]
      [Response Content-Type: application/json]
      [Response Validated: No]
```

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

```
hyperTextTransferProtocol 2
  OpenAPI Request
    Operation
    Request
    Response
      [Response Warning: Using guessed content-type: application/json]
      [Expert Info (Error/Response): Error]
      [Response Error: Validation: Value at root[authResult] does not match any entry in enumeration]
      [Response Headers Frame: 688]
      [Response Data Frame: 688]
      [Response Data: {"authResult":true,"kseaf":"7421637246dcc09523990ed67344a14da5bafc4d436ecbb802c8f0;"}]
      [Response Status: 200]
      [Response Content-Type: application/json]
      [Response Validated: No]
```

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