

Aufbau & Analyse von Industrie-Netzwerken

Lerneinheit 1: Arbeiten mit Wireshark

Prof. Dr. Christoph Karg

Studiengang Informatik
Hochschule Aalen



Sommersemester 2016



Gliederung

Diese Lerninheit beschäftigt sich mit Wireshark, einem leistungsfähigen Werkzeug zum Sniffing und zur Analyse von Rechnernetzen.

Sie gliedert sich in folgende Abschnitte:

1. Einführung in Wireshark
2. Das Calculation Protokoll
3. Die Programmiersprache Lua
4. Entwicklung eines Wireshark Plugins mit Lua

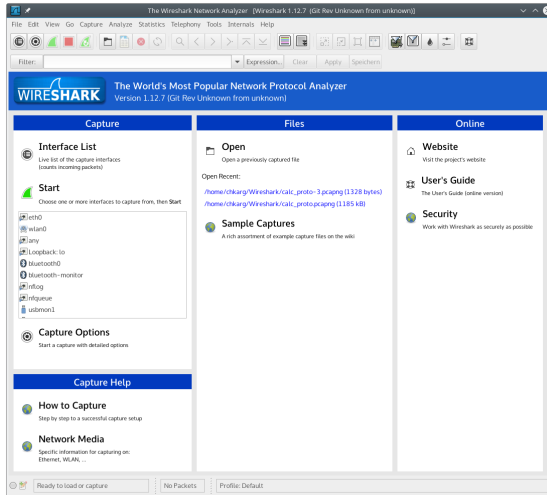
Wireshark

- Sammlung von Werkzeugen zum Netzwerk-Sniffing
- Für alle gängigen Betriebssysteme erhältlich
- GUI und Konsolenprogramme
- Bereitstellung zahlreicher Funktionen zur Netzwerkanalyse
- Unterstützung vieler Internet-Protokolle
- Erstellung von Paketfiltern auf Basis einer Filter-Sprache
- Durch Plugins erweiterbar
- Für weitere Informationen siehe: <https://www.wireshark.org>

In Wireshark enthaltene Werkzeuge

- Wireshark \rightsquigarrow GUI zur komfortablen Netzwerkanalyse
- TShark \rightsquigarrow Kommandozeilenprogramm zur Netzwerkanalyse
- dumpcap \rightsquigarrow Werkzeug zum Netzwerk-Sniffing
- mergecap \rightsquigarrow Werkzeug zum Vereinigen mehrerer Netzwerk-Mitschnitte zu einer Datei
- capinfos \rightsquigarrow Ausgabe von Informationen zu einem Netzwerk-Mitschnitt
- reordercap \rightsquigarrow Sortieren der Pakete eines Netzwerk-Mitschnitts anhand ihres Zeitstempels

Live Demo Wireshark GUI



Calculation Protokoll

- Protokoll zur Durchführung einfacher Berechnungen
- Ablauf:
 1. Der Client sendet einen Rechenauftrag an den Server.
 2. Der Server führt die Berechnung durch und sendet das Ergebnis zurück an den Client.
- Der Server “hört” auf den TCP-Port 10000.

Aufbau eines Calculation Pakets

Message ID

Nutzdaten

- Jedes Paket des Calculation Protokolls besteht aus:
 - ▷ Message ID \rightsquigarrow 1 Byte
 - ▷ Nutzdaten \rightsquigarrow variable Anzahl von Bytes
- Unterscheidung:
 - ▷ Calculation Request \rightsquigarrow Message ID = 1
 - ▷ Calculation Response \rightsquigarrow Message ID = 2

Calculation Request



- Nutzdaten eines Calculation Requests:
 1. Operator \rightsquigarrow Rechenoperation
 2. Zahl 1 \rightsquigarrow ganze Zahl
 3. Zahl 2 \rightsquigarrow ganze Zahl
- Mögliche Rechenoperationen:
 - ▷ Addition \rightsquigarrow Operator = 1
 - ▷ Subtraktion \rightsquigarrow Operator = 2
 - ▷ Multiplikation \rightsquigarrow Operator = 3
 - ▷ Division \rightsquigarrow Operator = 4

Kodierung einer ganzen Zahl

Länge	Vorzeichen	Betrag
-------	------------	--------

- Bestandteile:
 - ▷ Länge (2 Byte) \rightsquigarrow Anzahl der Bytes von Vorzeichen und Betrag
 - ▷ Vorzeichen (1 Byte) \rightsquigarrow Vorzeichen der Zahl
 - ▷ Betrag (Länge - 1 Bytes) \rightsquigarrow Betrag der Zahl
- Interpretation des Vorzeichen-Bytes:
 - ▷ Vorzeichen = 0 \rightsquigarrow Zahl ≥ 0
 - ▷ Vorzeichen = 1 \rightsquigarrow Zahl < 0
- Die Zahlen werden im Little Endian Format abgespeichert.

Kodierung einer ganzen Zahl – Beispiel

Beispiel: Kodierung der Zahl -123456 :

$[4, 0, 1, 64, 226, 1]$

Dekodierung:

- Länge: $4 + 256 \cdot 0 = 4$ Byte
- Vorzeichen: $1 \rightsquigarrow \text{Zahl} < 0$
- Betrag: $64 + 256 \cdot 226 + 256^2 \cdot 1 = 123456$

Ergebnis: -123456

Calculation Response

2

Zahl 1

- Nutzdaten eines Calculation Response:
 1. Zahl \rightsquigarrow ganze Zahl
- Die Zahl wird in obigem Format kodiert.

Mitschnitt eines Datenverkehrs

Ziel: Darstellung der Pakete des Calculation Protokolls in der Wireshark-GUI

Plan: Erstellung eines passenden Wireshark Plugins unter Einsatz der Programmiersprache Lua

Die Programmiersprache Lua

- Lua ist eine Entwicklung der Computer Graphics Technology Group (Tecgraf) der Pontifical Catholic University of Rio de Janeiro, Brasilien.
- Lua ist das portugiesische Wort für Mond.
- Die Entwicklung von Lua begann im Jahr 1993.
- Lua wurde für die Programmierung von Embedded Systemen entwickelt.
- Lua war von Anfang an als Skriptsprache konzipiert.
- Lua ist eine “leichtgewichtige” Sprache. Der entsprechende Interpreter ist etwa 195 KByte groß.
- Weitere Informationen findet man unter <https://www.lua.org>.

LUA in der Praxis

- Lua wird in der Praxis in vielen Projekten eingesetzt.
- Projektauswahl:
 - ▷ Adobe Lightroom
 - ▷ L^AT_EX
 - ▷ CryEngine
 - ▷ Wireshark
 - ▷ ...

Hello World! in Lua

```
1  -- Hello World
2  print("Hello_World")
```

Berechnung der Fakultätsfunktion

```
1  -- defines a factorial function
2  function fact (n)
3      if n == 0 then
4          return 1
5      else
6          return n * fact(n-1)
7      end
8  end
9
10 print("enter a number:")
11 -- read a number from the terminal
12 a = io.read("*number")
13 print(fact(a))
```


Wireshark & Lua

- Wireshark setzt Lua als Skriptsprache für Plugins ein.
- Lua ist für Rapid Prototyping von Wireshark Erweiterungen gedacht.
- Die Ausführung von in Lua entwickelten Plugins ist vergleichsweise langsam, da der Code interpretiert wird.
- Die in Wireshark integrierten Netzwerk-Filter sind in C programmiert.

Wireshark Lua API

- Das Wireshark Online Manual [LOB14a] widmet der Lua API ein eigenes Kapitel [LOB14b].
- Wichtige Komponenten sind:
 - ▷ Proto \rightsquigarrow Anlegen eines neuen Protokolls
 - ▷ PInfo \rightsquigarrow Informationen über das aktuelle Paket
 - ▷ Tvb \rightsquigarrow Buffer mit dem Inhalt des aktuellen Pakets
 - ▷ TvbRange \rightsquigarrow Ein Bereich innerhalb des Tvb
 - ▷ TreeItem \rightsquigarrow Knoten im Darstellungsbaum der GUI

Wireshark Dissectors

- Die Aufgabe eines Dissectors ist die Analyse eines Teils von einem Datenpaket.
- Ein Dissector muss unter Verwendung von Proto für ein Transportprotokoll und einen Port registriert werden.
- Dissectoren werden automatisch ausgeführt, wenn sie einem Protokoll zugeordnet sind. Alternativ kann man sie händisch mittels des Menüpunkts “Decode As” starten.
- Beim Aufruf erhält ein Dissector einen TVB Buffer, ein PlInfo Objekt und ein Treeltem Objekt.

Praktischer Teil

Praktischer Teil: Entwicklung eines Dissectors für das Calculation Protokoll

Literatur I

- [LOB14a] Ulf Lamping, Luis E. Ontanon und Graham Bloice. *Wireshark Developer's Guide For Wireshark 2.1*. 2014. URL: https://www.wireshark.org/docs/wsdg_html_chunked/ (besucht am 12.04.2016).
- [LOB14b] Ulf Lamping, Luis E. Ontanon und Graham Bloice. *Wireshark's Lua API Reference Manual*. 2014. URL: https://www.wireshark.org/docs/wsdg_html_chunked/wsluarm_modules.html (besucht am 12.04.2016).