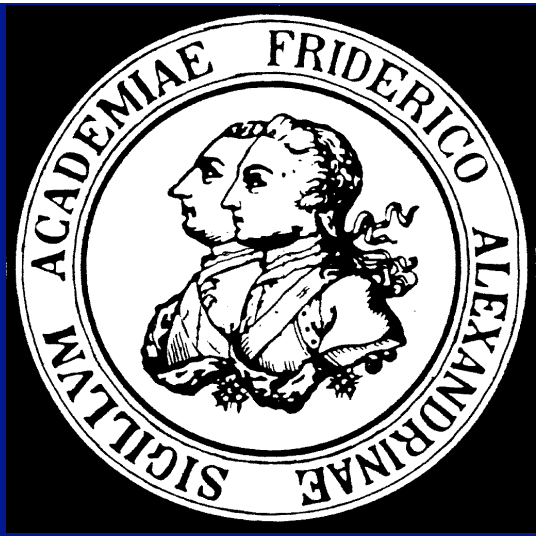
14.07.2010

Julian Reisser
Julian.Reisser@ce.stud.uni-erlangen.de



KvBK: Basic Authentication, Digest Authentication, OAuth

Motivation

- Authentifizierung
  - Nachweis, dass man der ist für den man sich ausgibt
  - Durch Austausch eines Geheimnisses oder Besitzes
- Autorisierung
  - Das Recht bestimmte Ressourcen nutzen zu dürfen

Motivation

- World Wide Web
  - Basierend auf HTTP
  - Meistgenutzte Komponente des Internets
  - Unzählige Webseiten bieten öffentliche oder private, kostenfreie oder kostenpflichtige Dienste an
  - ➢ Authentifizierung und Autorisierung ist nötig

Gliederung

- 1 Basic Authentication
  - 1.1 Aufbau
  - 1.2 Beispiel mit Wireshark
- 2 Digest Authentication
  - 2.1 In einem Satz
  - 2.2 Aufbau
- 3 OAuth
  - 3.1 Begriffsklärung
  - 3.2 Einführendes Beispiel
  - 3.3 Aufbau

1 Basic Authentication

- Standardimplementierung der Authentifizierung über HTTP
- Teil des Hypertext Transfer Protokoll
- Trotz Schwächen immer noch häufig gebraucht

1.1 Aufbau

- Schritt 1: Bei Zugriff auf Datei antwortet Webserver
  - `401 Unauthorized`
    `[…]`
    `WWW-Authenticate: Basic realm="abc"`
    `[…]`

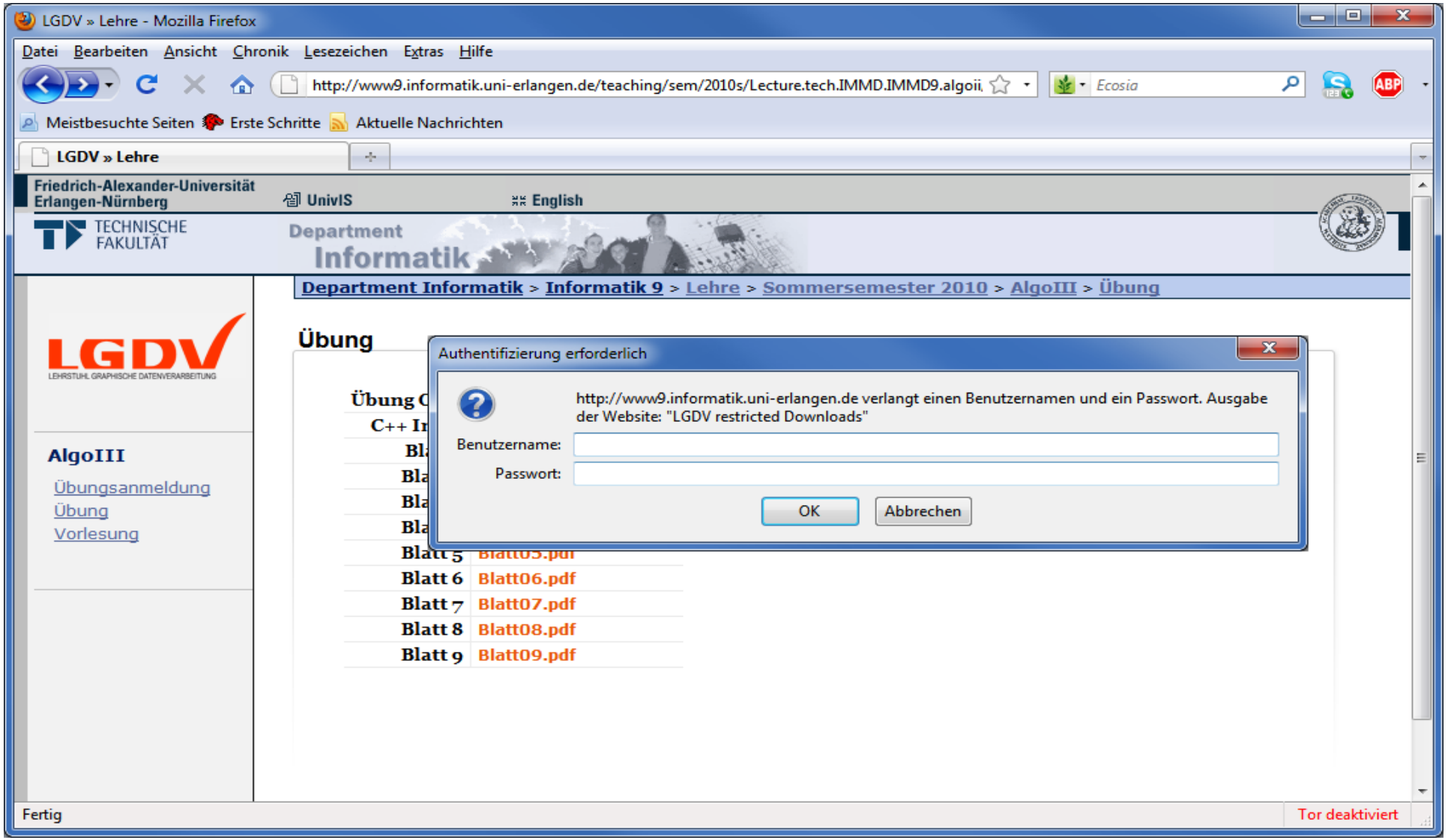
    - `realm`: Name des geschützten Bereichs

1.1 Aufbau

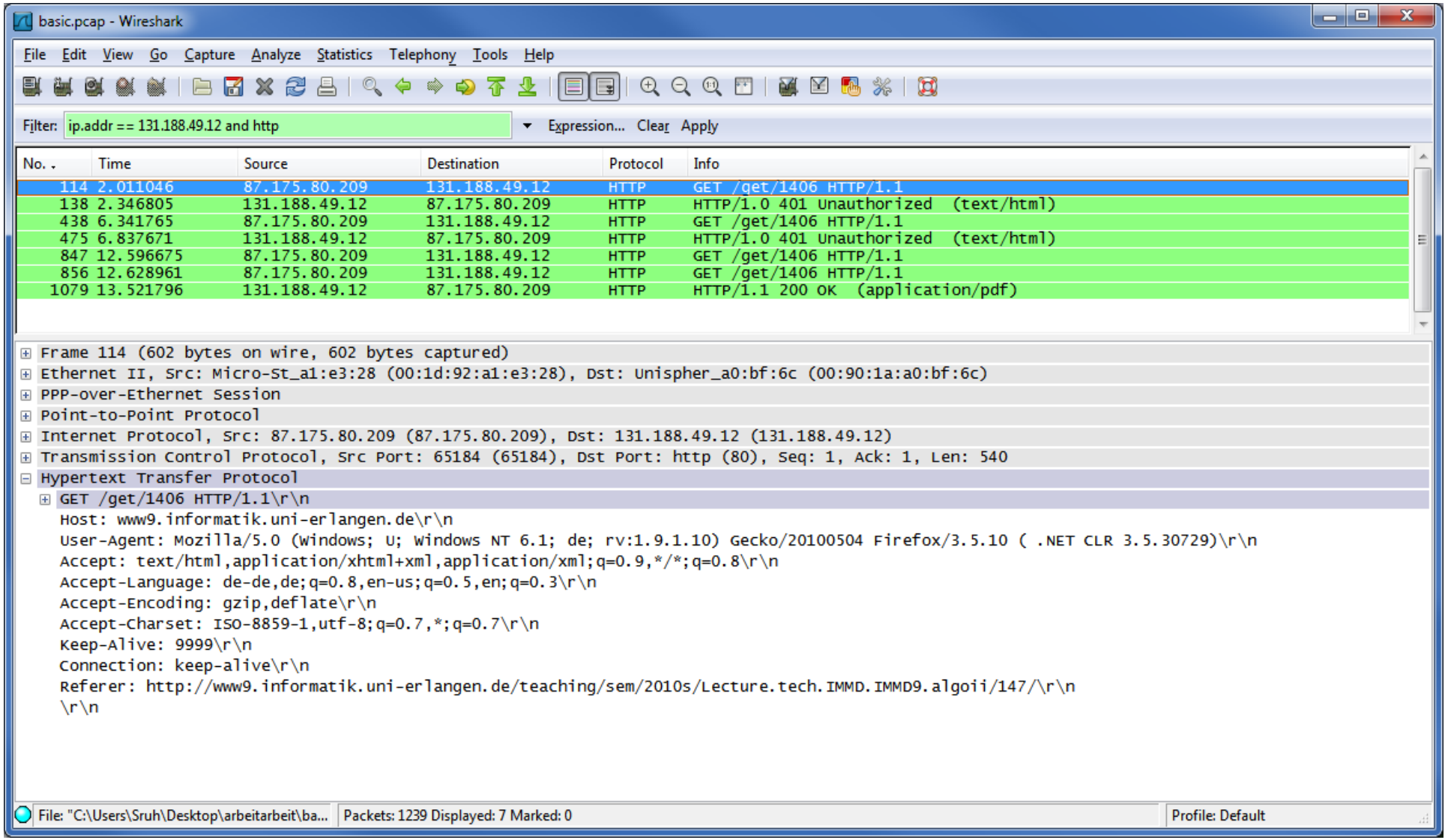
- Schritt 2: Webbrowser fordert Passworteingabe
- Schritt 3: Erneuter Versuch, mit zusätzlicher Zeile im Header
  - `GET /get/index`
    `[…]`
    `Authorization: Basic S3ZCSzpLdkJL`
    `[…]`

    - enthält Daten der Form Benutzername:Passwort
    - `S3ZCSzpLdkJL` ist Base64 codiert für KvBK:KvBK

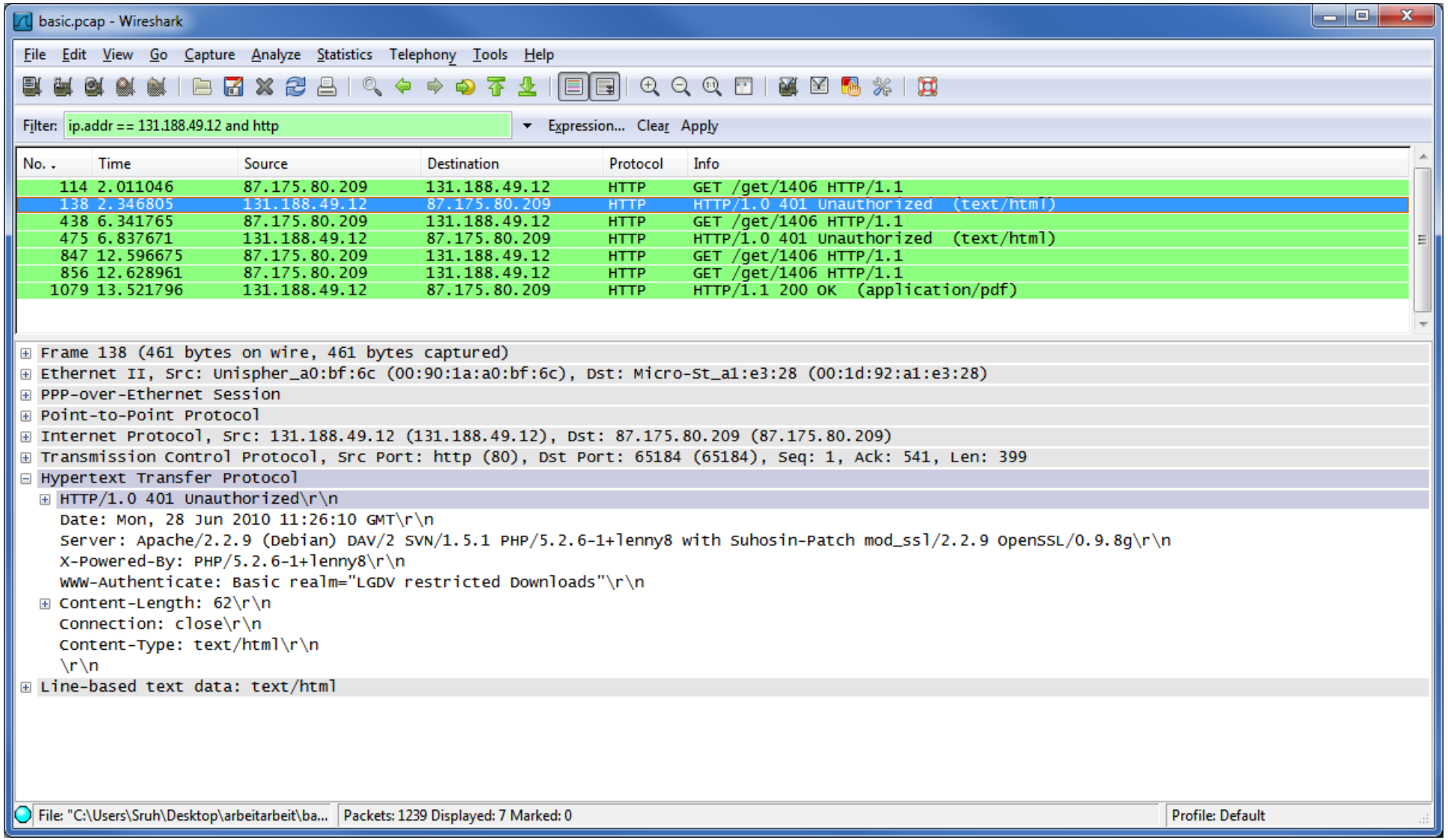
1.2 Beispiel mit Wireshark



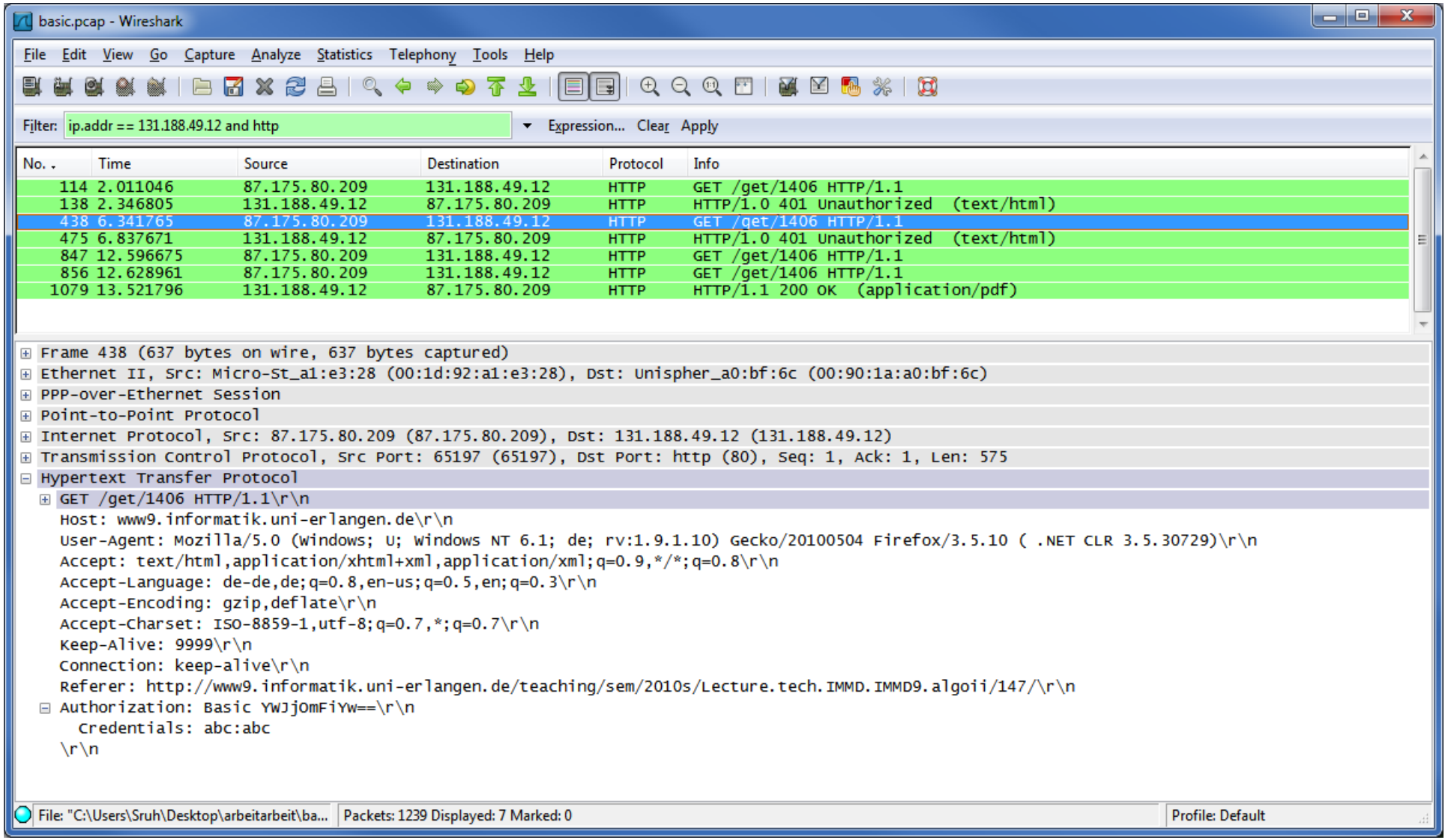
1.2 Beispiel mit Wireshark



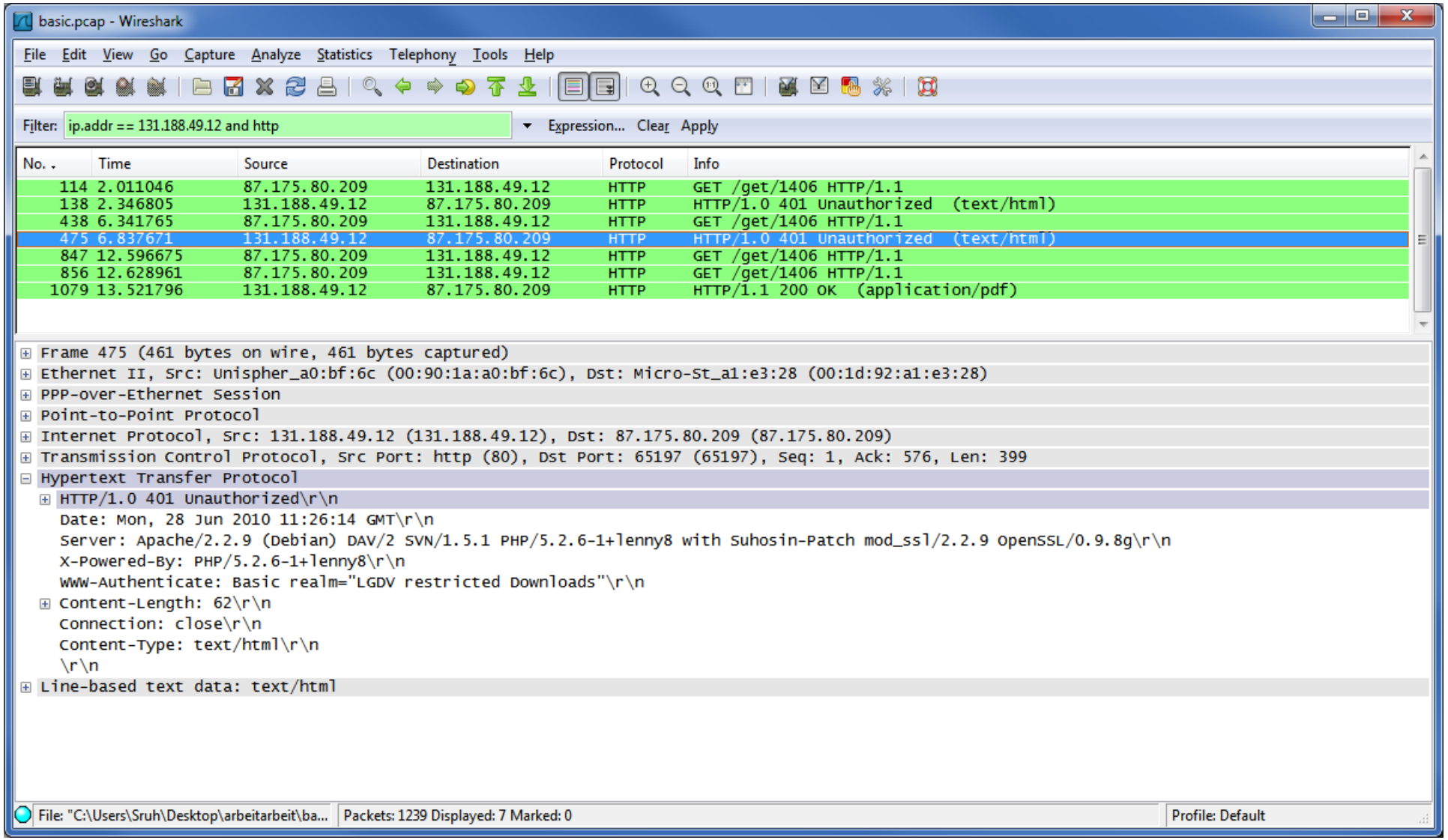
1.2 Beispiel mit Wireshark



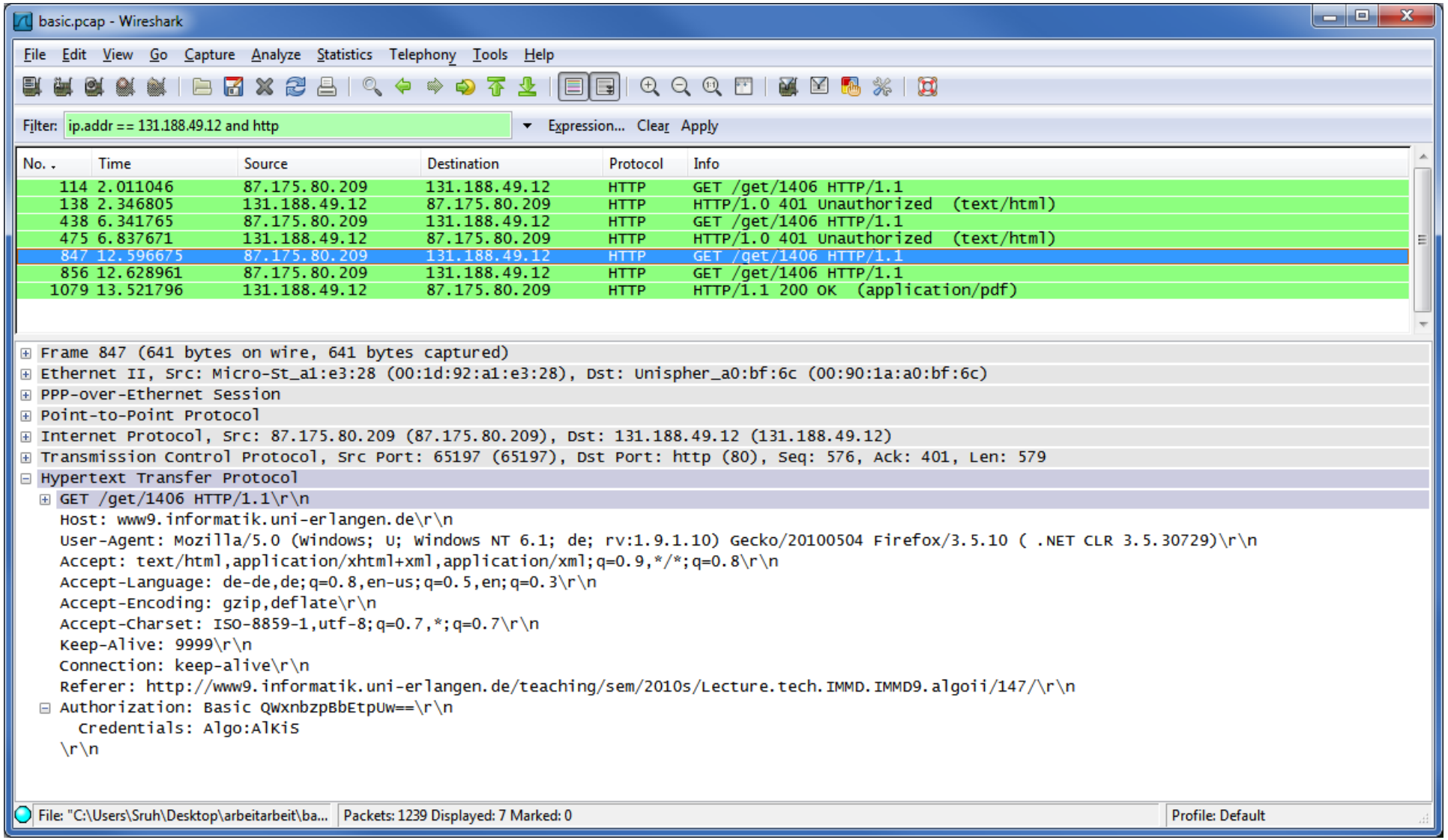
1.2 Beispiel mit Wireshark



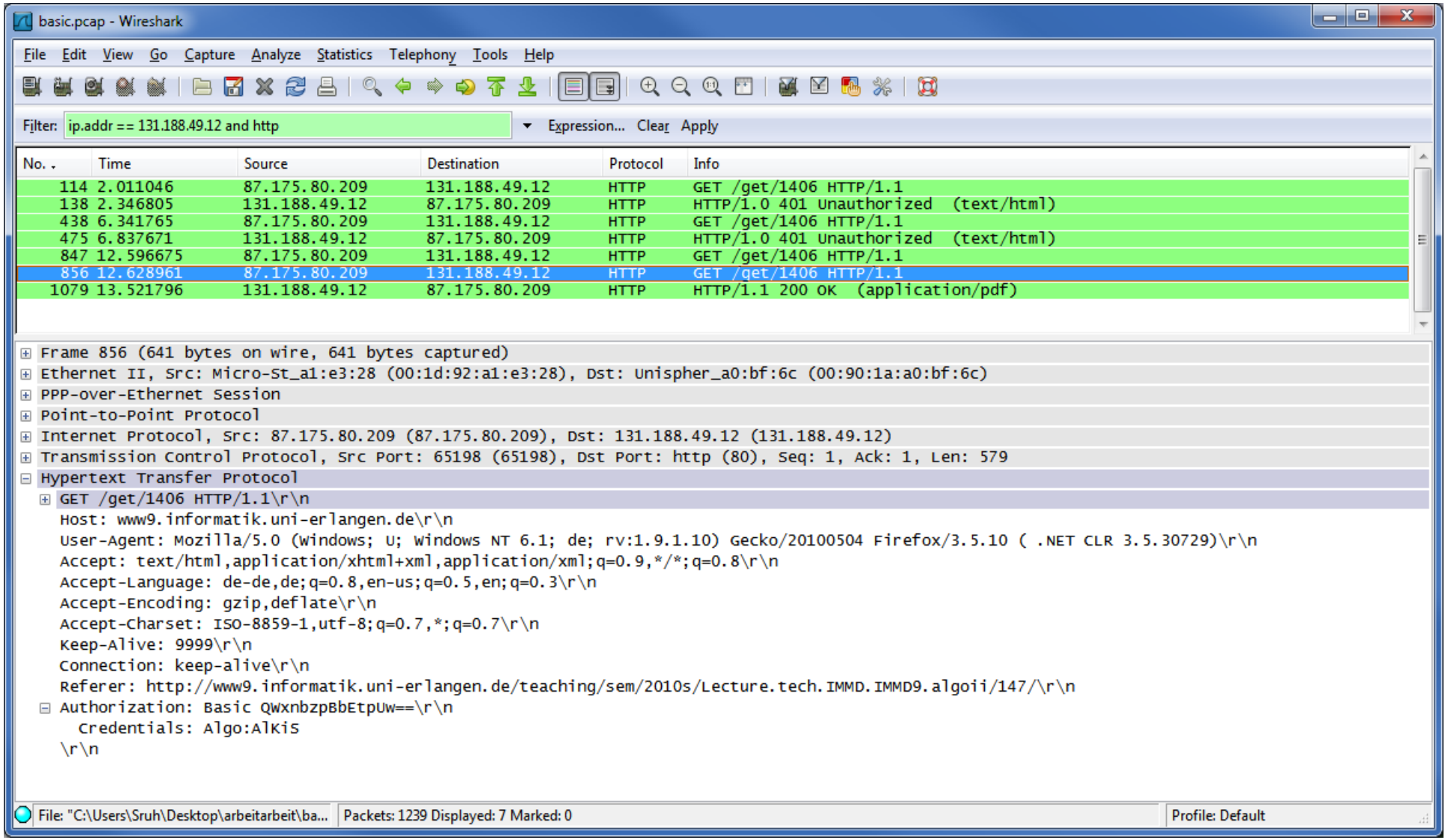
1.2 Beispiel mit Wireshark



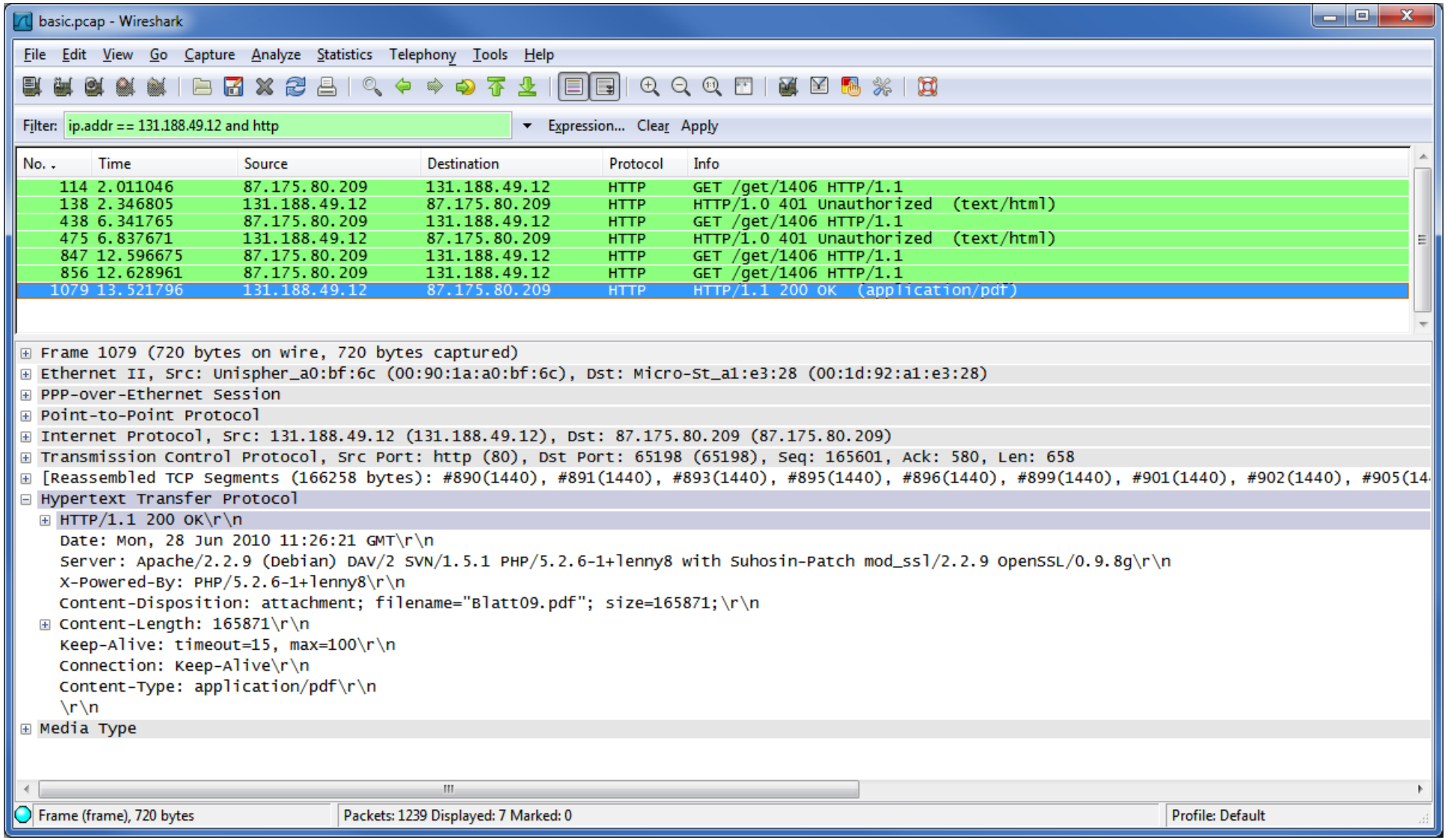
1.2 Beispiel mit Wireshark



1.2 Beispiel mit Wireshark



1.2 Beispiel mit Wireshark



2 Digest Authentication

- Basic Authentication unsicher, da Benutzername und Passwort nur codiert, nicht aber verschlüsselt übermittelt wird
- Digest als sicherer Nachfolger

2.1 In einem Satz

- Im Unterschied zu Basic, sendet Digest Authentication Benutzername und Passwort nicht direkt, sondern lediglich den Benutzernamen und einen Hashwert aus Passwort, einer Zufallszahl und anderen Variablen
- Webserver kann Hashwert seinerseits berechnen

2.2 Aufbau

- Aufbau ähnlich wie Basic
  - `WWW-Authenticate`: Informationen des Webservers über den geschützten Bereich, u.a. Zufallszahl
  - `Authorization`: Passwortübermittlung des Benutzers um Zugriff zu erlangen
  - `Authentication-Info`: Extra Headerzeile des Webservers, um Zufallszahl nach jeder Anfrage zu ändern

2.2 Aufbau

- WWW-Authenticate Headerzeile
  - `WWW-Authenticate: Digest`
    `realm = "",`
    `domain = "",`
    `nonce = "",`
    `algorithm = "",`
    `qop-options = ""`

    - `realm`: Name des geschützten Bereichs
    - `domain`: Liste von URIs für welche der Zugriff gilt
    - `nonce`: Zufallszahl am besten mit Zeitstempel (Replay-Angriff)
    - `algorithm`: Mögliche Angabe der verwendeten Hashfunkion
      Standard: "MD5“
    - `qop-options`: "quality of protection“: Möglichkeit für den Benutzer mehr Sicherheitsmechanismen zu aktivieren

2.2 Aufbau

- Authorization Headerzeile
  - `Authorization: Digest`

```
username =    "",  // Benutzername
realm =       "",  // Name des Bereichs
nonce =       "",  // wie vom Server
algorithm =   "",  // Gewählter Algorithmus
response =    "",  // Der Hashwert aus Passwort, nonce und mehr
message-qop ="",   // Gewählte qop
nonce-count ="",   // Anzahl an Paketen die mit dieser nonce
                   // angefordert wurden (Aktiviert mit qop)
cnonce =      ""   // Zufallszahl des Benutzers (Aktiviert mit qop)
```

2.2 Aufbau

- Berechnung des Hashwerts
  - Aufbau des Strings `value`
    - `value = concat(username, passwd, […], ":", nonce, […])`
  - Hashen des Strings mit gewähltem Algorithmus `algorithm (value)`
  - Je nachdem wie `qop` gewählt gegebenenfalls noch `qop`, `cnonce`, `nonce-count` in `value`

2.2 Aufbau

- Authentication-Info Headerzeile
  - Authentication-Info

```
nextnonce =   "",     // Für nächste Anfrage
message-qop = "",
cnonce =      "",
nonce-count = ""
```

3 OAuth

- Offenes Protokoll, das eine standardisierte, sichere Autorisierung für Desktop-, Web- und mobile Anwendungen über HTTP erlaubt
- Erste Version 2007 von Blaine Cook und Chris Messina

3.1 Begriffsklärung

- Endnutzer: Der, dem das ganze was bringen soll
- Service Provider: HTTP Server der Ressourcen für andere bereitstellt
- Konsument: Dienst der auf Ressourcen von Service Providern zugreift
- Token
  - Anfragetoken: Ein Token, das nachdem es vom Endbenutzer verifiziert wurde gegen ein Zugriffstoken getauscht werden kann
  - Zugriffstoken: Ermöglicht Zugriff auf Ressourcen eines Service Providers

3.2 Einführendes Beispiel

Service Provider: Photo Dienst (flickr.com)

- Photos hochladen
- Benutzer/Passwort eingeben und Anfrage von plaxo bestätigen, Endbenutzer zu plaxo weiterleiten

Konsument: Druck Dienst (plaxo.com)

- Photos von flickr drucken
- Holt sich Anfragetoken von flickr, und leitet Endnutzer zur verifikation zu flickr weiter
- Kann Anfragetoken gegen Zugriffstoken tauschen und Photos laden

3.3 Aufbau

- Konsument bekommt von Service Provider einmalig
  - `consumer_key`
  - `shared_secret`
  - URL für Anfragetoken
    - z.B. https://Flickr.com/anfragetoken
  - URL wo der Endbenutzer Anfragetoken verifizieren kann
    - z.B. https://Flickr.com/authorize
  - URL für Zugriffstoken
    - z.B. https://Flickr.com/zugriffstoken

3.3 Aufbau

- Anfragetoken holen
  - POST /Anfragetoken HTTP/1.1

```
Host: Flickr.com
Authorization: OAuth
realm =                    "",
oauth_consumer_key =       "",
oauth_signature_method = "", // "HMAC-SHA1"
oauth_timestamp =          "",
oauth_nonce =              "", // Analog zu Digest
oauth_signature=           ""  // Analog zu Digest
```

3.3 Aufbau

- Anfragetoken bekommen
  - ```
    HTTP/1.1 200 OK
    Content-Type: application/x-www-form-
                          urlencoded
    oauth_token =          "",
    oauth_token_secret = ""
    ```

- URL für Endbenutzer zum verifizieren https://Flickr.com/authorize?oauth_token=`oauth_token`
- URL um Endnutzer zurück zu plaxo zu schicken http://Plaxo.com/ready?oauth_token=`oauth_token`
    ```

3.3 Aufbau

- Zugriffstoken über TLS holen
  - POST /Zugriffstoken HTTP/1.1

```
Host: Plaxo.com
Authorization: OAuth
realm =                    "",
oauth_consumer_key =       "",
oauth_token =              "", // das Anfragetoken
oauth_signature_method =   "",
oauth_timestamp=           "",
oauth_nonce =              "",
oauth_signature =          ""
```

Danke für die Aufmerksamkeit

3.3 Aufbau

- Zugriffstoken bekommen
  - HTTP/1.1 200 OK

```
Content-Type: application/x-www-form-
              urlencoded
oauth_token = "", // das Zugriffstoken
oauth_token_secret = ""
```

3.3 Aufbau

- Photos von flickr holen
  - GET /photos?file=vacation.jpg&size=original HTTP/1.1

```
Host: Flickr.com
Authorization: OAuth
      realm =                    "",
      oauth_consumer_key =       "",
      oauth_token =              "",
      oauth_signature_method =   "",
      oauth_timestamp =          "",
      oauth_nonce =              "",
      oauth_signature =          ""
```

Quellen

- http://www.ietf.org/rfc/rfc2617.txt
- http://www.heise.de/developer/artikel/Autorisierungsdienste-mit-OAuth-845382.html
- http://hueniverse.com/oauth/
- http://tools.ietf.org/html/rfc5849
- http://www.wireshark.org/
- http://www9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii/147/

Fragen?