

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

- [...]

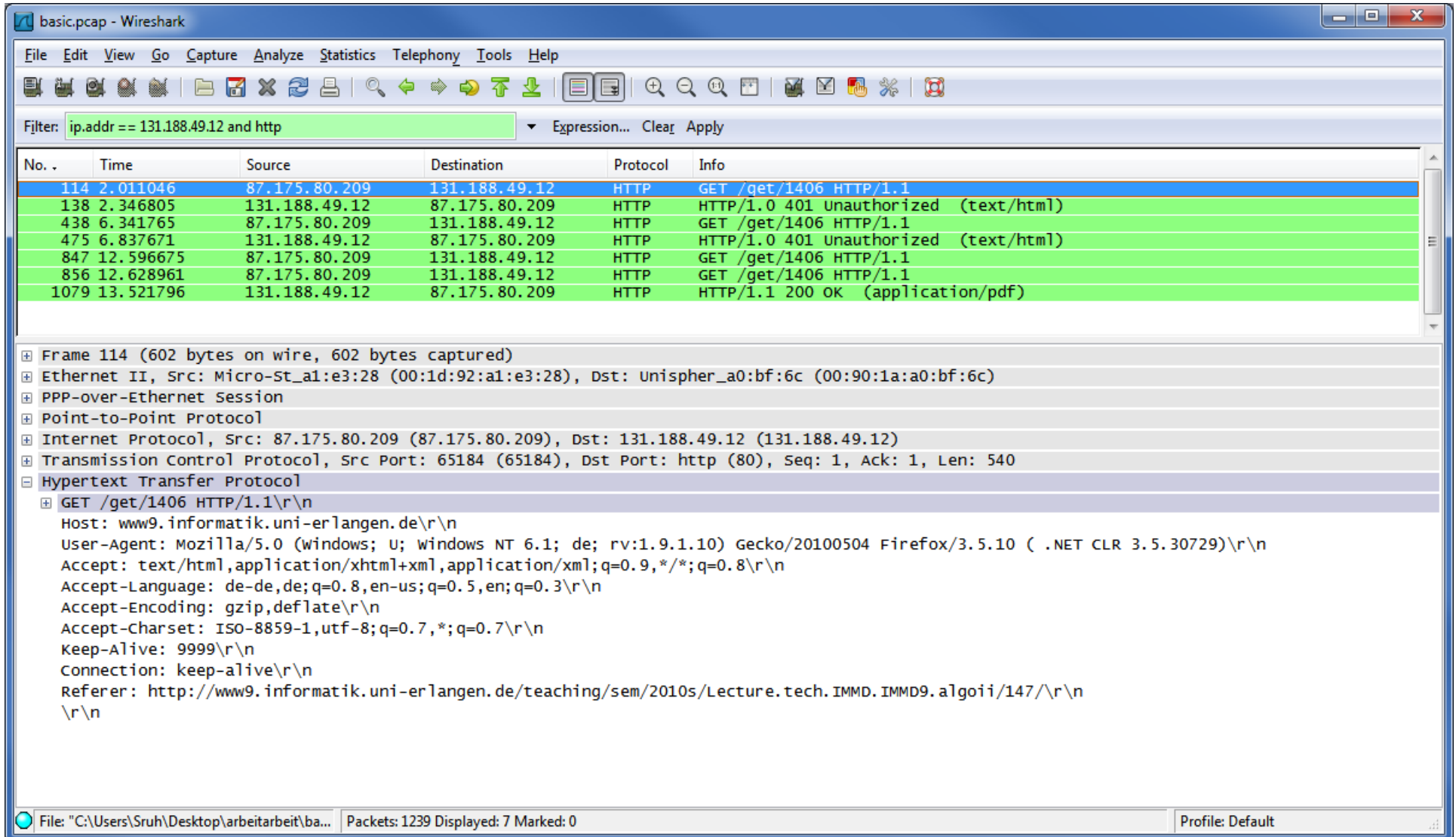
- enthält Daten der Form Benutzername:Passwort
 - S3ZCSzpLdkJL ist Base64 codiert für KvBK:KvBK

1.2 Beispiel mit Wireshark

The screenshot shows a Mozilla Firefox browser window with the following details:

- Address Bar:** `http://www9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii`
- Page Title:** LGDV » Lehre
- Page Content:** Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik, Sommersemester 2010, AlgoIII > Übung. A list of assignments (Blatt 5 to Blatt 9) is visible, each with a PDF link.
- Dialog Box:** A modal dialog titled "Authentifizierung erforderlich" (Authentication required) is displayed. It contains a question mark icon and the message: "http://www9.informatik.uni-erlangen.de verlangt einen Benutzernamen und ein Passwort. Ausgabe der Website: 'LGDV restricted Downloads'". Below the message are input fields for "Benutzername:" and "Passwort:", and "OK" and "Abbrechen" buttons.
- Page Footer:** "Fertig" and "Tor deaktiviert" are visible at the bottom.

1.2 Beispiel mit Wireshark



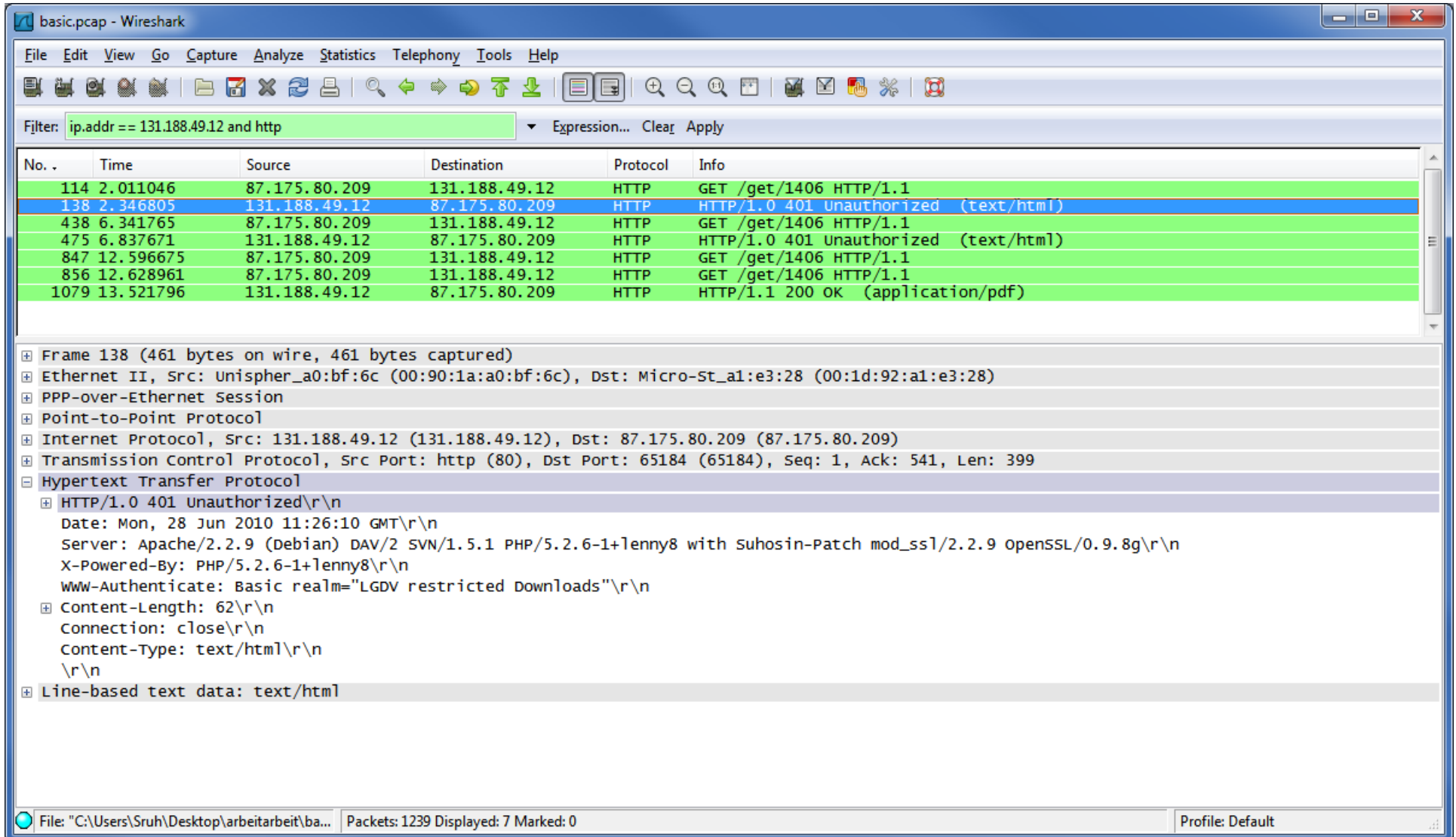
The screenshot shows the Wireshark interface with a packet capture of an HTTP GET request and response. The filter is set to `ip.addr == 131.188.49.12 and http`. The packet list shows several packets, with packet 114 selected. The packet details pane shows the structure of the selected packet, including Ethernet II, PPP-over-Ethernet Session, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol.

No. .	Time	Source	Destination	Protocol	Info
114	2.011046	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
138	2.346805	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
438	6.341765	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
475	6.837671	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
847	12.596675	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
856	12.628961	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
1079	13.521796	131.188.49.12	87.175.80.209	HTTP	HTTP/1.1 200 OK (application/pdf)

Frame 114 (602 bytes on wire, 602 bytes captured)
Ethernet II, Src: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28), Dst: unispher_a0:bf:6c (00:90:1a:a0:bf:6c)
PPP-over-Ethernet Session
Point-to-Point Protocol
Internet Protocol, Src: 87.175.80.209 (87.175.80.209), Dst: 131.188.49.12 (131.188.49.12)
Transmission Control Protocol, Src Port: 65184 (65184), Dst Port: http (80), Seq: 1, Ack: 1, Len: 540
Hypertext Transfer Protocol
GET /get/1406 HTTP/1.1\r\n
Host: ww9.informatik.uni-erlangen.de\r\n
User-Agent: Mozilla/5.0 (windows; u; windows NT 6.1; de; rv:1.9.1.10) Gecko/20100504 Firefox/3.5.10 (.NET CLR 3.5.30729)\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 9999\r\n
Connection: keep-alive\r\n
Referer: http://ww9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii/147/\r\n\r\n

File: "C:\Users\Sruh\Desktop\arbeit\ba... Packets: 1239 Displayed: 7 Marked: 0 Profile: Default

1.2 Beispiel mit Wireshark



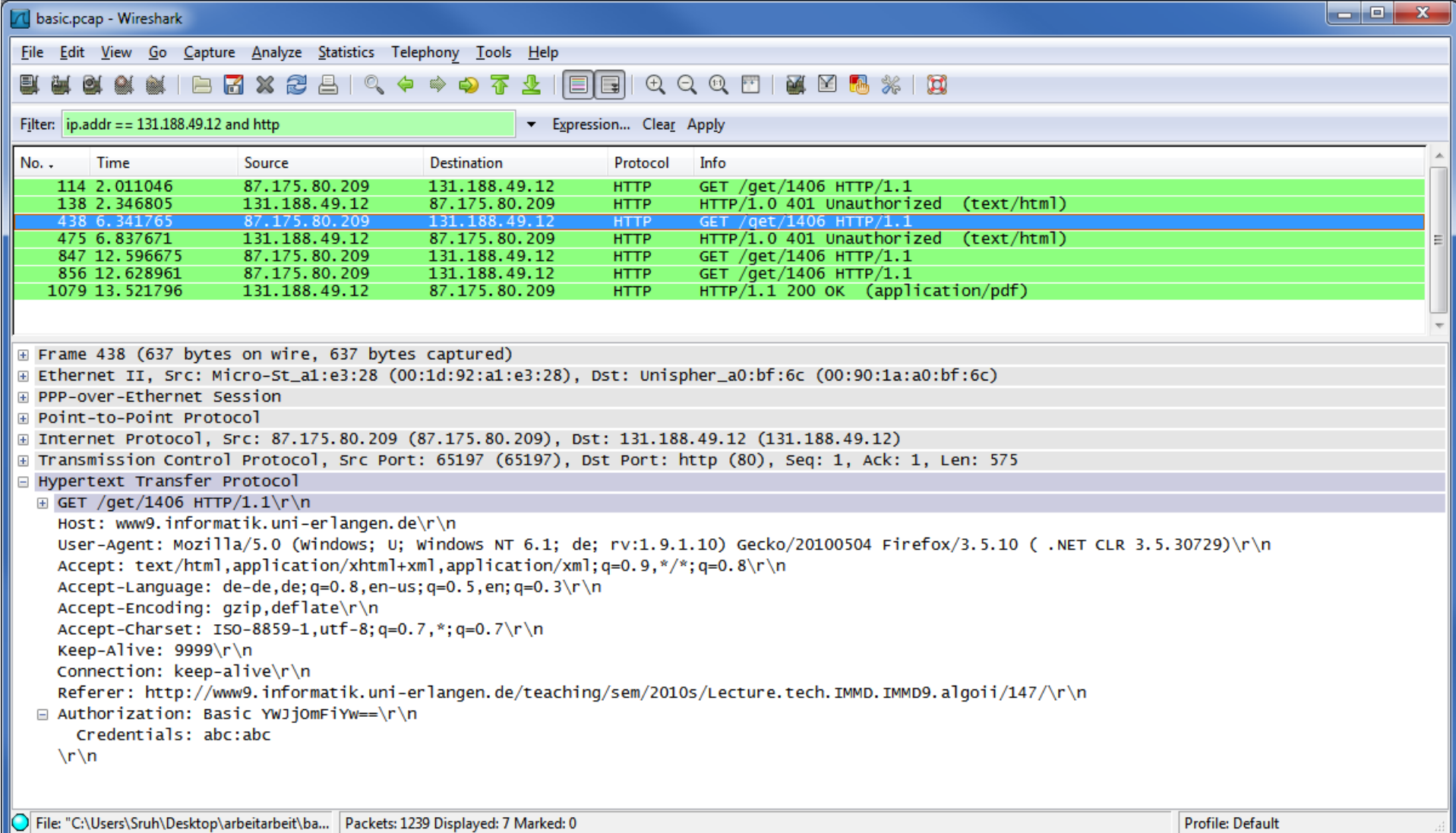
The screenshot shows the Wireshark interface with a packet capture of an HTTP 401 Unauthorized response. The filter is set to `ip.addr == 131.188.49.12 and http`. The packet list shows several HTTP requests and responses, with packet 138 highlighted in blue, indicating it is the selected packet. The packet details pane shows the structure of the selected packet, including Ethernet II, PPP-over-Ethernet Session, Point-to-Point Protocol, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The Hypertext Transfer Protocol section is expanded, showing the response status and headers.

No. .	Time	Source	Destination	Protocol	Info
114	2.011046	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
138	2.346805	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
438	6.341765	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
475	6.837671	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
847	12.596675	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
856	12.628961	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
1079	13.521796	131.188.49.12	87.175.80.209	HTTP	HTTP/1.1 200 OK (application/pdf)

Frame 138 (461 bytes on wire, 461 bytes captured)
Ethernet II, Src: unispher_a0:bf:6c (00:90:1a:a0:bf:6c), Dst: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28)
PPP-over-Ethernet Session
Point-to-Point Protocol
Internet Protocol, Src: 131.188.49.12 (131.188.49.12), Dst: 87.175.80.209 (87.175.80.209)
Transmission Control Protocol, Src Port: http (80), Dst Port: 65184 (65184), Seq: 1, Ack: 541, Len: 399
Hypertext Transfer Protocol
HTTP/1.0 401 Unauthorized\r\n
Date: Mon, 28 Jun 2010 11:26:10 GMT\r\nServer: Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 PHP/5.2.6-1+lenny8 with Suhosin-Patch mod_ssl/2.2.9 openssl/0.9.8g\r\nX-Powered-By: PHP/5.2.6-1+lenny8\r\nwww-Authenticate: Basic realm="LGDV restricted Downloads"\r\n
Content-Length: 62\r\nConnection: close\r\nContent-Type: text/html\r\n\r\n
Line-based text data: text/html

File: "C:\Users\Sruh\Desktop\arbeit\ba... Packets: 1239 Displayed: 7 Marked: 0 Profile: Default

1.2 Beispiel mit Wireshark



The screenshot shows the Wireshark interface with a packet capture of an HTTP GET request and response. The filter is set to `ip.addr == 131.188.49.12 and http`. The packet list shows several packets, with packet 438 selected. The packet details pane shows the following information:

- Frame 438 (637 bytes on wire, 637 bytes captured)
- Ethernet II, Src: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28), Dst: unispher_a0:bf:6c (00:90:1a:a0:bf:6c)
- PPP-over-Ethernet Session
- Point-to-Point Protocol
- Internet Protocol, Src: 87.175.80.209 (87.175.80.209), Dst: 131.188.49.12 (131.188.49.12)
- Transmission Control Protocol, Src Port: 65197 (65197), Dst Port: http (80), Seq: 1, Ack: 1, Len: 575
- Hypertext Transfer Protocol
 - GET /get/1406 HTTP/1.1\r\n
 - Host: ww9.informatik.uni-erlangen.de\r\n
 - User-Agent: Mozilla/5.0 (windows; u; windows NT 6.1; de; rv:1.9.1.10) Gecko/20100504 Firefox/3.5.10 (.NET CLR 3.5.30729)\r\n
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
 - Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3\r\n
 - Accept-Encoding: gzip,deflate\r\n
 - Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
 - Keep-Alive: 9999\r\n
 - Connection: keep-alive\r\n
 - Referer: http://ww9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii/147/\r\n
 - Authorization: Basic YWJjOmFiYW==\r\n
 - Credentials: abc:abc

The status bar at the bottom shows: File: "C:\Users\Sruh\Desktop\arbeit\ba... Packets: 1239 Displayed: 7 Marked: 0 Profile: Default

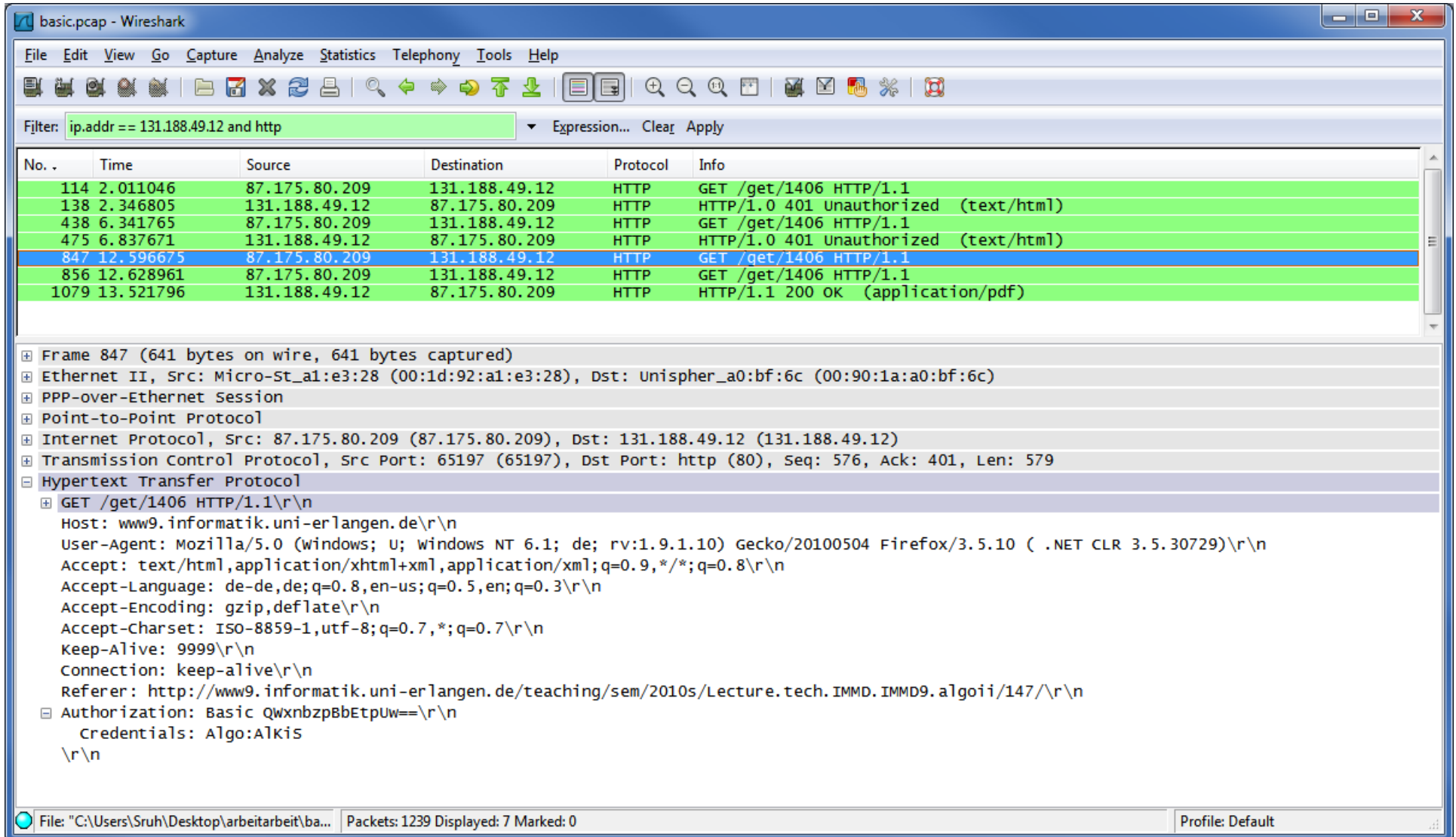
1.2 Beispiel mit Wireshark

The screenshot shows the Wireshark interface with a packet capture of an HTTP 401 Unauthorized response. The filter is set to `ip.addr == 131.188.49.12 and http`. The packet list shows several HTTP requests and responses, with packet 475 selected. The packet details pane shows the structure of the selected packet, including Ethernet II, PPP-over-Ethernet Session, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP response body contains the following text:

```
HTTP/1.0 401 Unauthorized\r\nDate: Mon, 28 Jun 2010 11:26:14 GMT\r\nServer: Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 PHP/5.2.6-1+lenny8 with Suhosin-Patch mod_ssl/2.2.9 openssl/0.9.8g\r\nX-Powered-By: PHP/5.2.6-1+lenny8\r\nWWW-Authenticate: Basic realm="LGDV restricted Downloads"\r\nContent-Length: 62\r\nConnection: close\r\nContent-Type: text/html\r\n\r\nLine-based text data: text/html
```

The status bar at the bottom indicates the file path, packet count (1239), displayed packets (7), and marked packets (0).

1.2 Beispiel mit Wireshark



The screenshot shows the Wireshark interface with a packet capture of an HTTP GET request and its response. The filter is set to `ip.addr == 131.188.49.12 and http`. The packet list shows several packets, with packet 847 selected. The packet details pane shows the structure of the selected packet, including Ethernet II, PPP-over-Ethernet Session, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP request is a GET for `/get/1406` from `www9.informatik.uni-erlangen.de`. The response is a 200 OK with `application/pdf` content type.

No. .	Time	Source	Destination	Protocol	Info
114	2.011046	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
138	2.346805	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
438	6.341765	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
475	6.837671	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
847	12.596675	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
856	12.628961	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
1079	13.521796	131.188.49.12	87.175.80.209	HTTP	HTTP/1.1 200 OK (application/pdf)

Frame 847 (641 bytes on wire, 641 bytes captured)
Ethernet II, Src: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28), Dst: unispher_a0:bf:6c (00:90:1a:a0:bf:6c)
PPP-over-Ethernet Session
Point-to-Point Protocol
Internet Protocol, Src: 87.175.80.209 (87.175.80.209), Dst: 131.188.49.12 (131.188.49.12)
Transmission Control Protocol, Src Port: 65197 (65197), Dst Port: http (80), Seq: 576, Ack: 401, Len: 579
Hypertext Transfer Protocol
GET /get/1406 HTTP/1.1\r\nHost: www9.informatik.uni-erlangen.de\r\nUser-Agent: Mozilla/5.0 (windows; u; windows NT 6.1; de; rv:1.9.1.10) Gecko/20100504 Firefox/3.5.10 (.NET CLR 3.5.30729)\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\nKeep-Alive: 9999\r\nConnection: keep-alive\r\nReferer: http://www9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii/147/\r\nAuthorization: Basic qwxnbzpBbEtpUw==\r\nCredentials: Algo:Alkis\r\n\r\n

File: "C:\Users\Sruh\Desktop\arbeit\ba... Packets: 1239 Displayed: 7 Marked: 0 Profile: Default

1.2 Beispiel mit Wireshark

The screenshot shows the Wireshark interface with a filter set to `ip.addr == 131.188.49.12 and http`. The packet list pane displays several packets, with packet 856 selected. The packet details pane shows the structure of the selected packet, including Ethernet II, PPP-over-Ethernet Session, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP request details are expanded, showing the request line, headers, and authorization information.

No. .	Time	Source	Destination	Protocol	Info
114	2.011046	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
138	2.346805	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
438	6.341765	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
475	6.837671	131.188.49.12	87.175.80.209	HTTP	HTTP/1.0 401 Unauthorized (text/html)
847	12.596675	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
856	12.628961	87.175.80.209	131.188.49.12	HTTP	GET /get/1406 HTTP/1.1
1079	13.521796	131.188.49.12	87.175.80.209	HTTP	HTTP/1.1 200 OK (application/pdf)

Frame 856 (641 bytes on wire, 641 bytes captured)
Ethernet II, Src: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28), Dst: unispher_a0:bf:6c (00:90:1a:a0:bf:6c)
PPP-over-Ethernet Session
Point-to-Point Protocol
Internet Protocol, Src: 87.175.80.209 (87.175.80.209), Dst: 131.188.49.12 (131.188.49.12)
Transmission Control Protocol, Src Port: 65198 (65198), Dst Port: http (80), Seq: 1, Ack: 1, Len: 579
Hypertext Transfer Protocol
GET /get/1406 HTTP/1.1\r\n
Host: www9.informatik.uni-erlangen.de\r\n
User-Agent: Mozilla/5.0 (windows; u; windows NT 6.1; de; rv:1.9.1.10) Gecko/20100504 Firefox/3.5.10 (.NET CLR 3.5.30729)\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 9999\r\n
Connection: keep-alive\r\n
Referer: http://www9.informatik.uni-erlangen.de/teaching/sem/2010s/Lecture.tech.IMMD.IMMD9.algoii/147/\r\n
Authorization: Basic qwxnbzpBbEtpUw==\r\n
Credentials: Algo:Alkis
\r\n

File: "C:\Users\Sruh\Desktop\arbeit\ba... Packets: 1239 Displayed: 7 Marked: 0 Profile: Default

1.2 Beispiel mit Wireshark

The screenshot shows the Wireshark interface with a filter set to `ip.addr == 131.188.49.12 and http`. The packet list pane shows several HTTP requests and one response (No. 1079). The packet details pane for frame 1079 shows the following structure:

- Frame 1079 (720 bytes on wire, 720 bytes captured)
- Ethernet II, Src: unispher_a0:bf:6c (00:90:1a:a0:bf:6c), Dst: Micro-st_a1:e3:28 (00:1d:92:a1:e3:28)
- PPP-over-Ethernet Session
- Point-to-Point Protocol
- Internet Protocol, Src: 131.188.49.12 (131.188.49.12), Dst: 87.175.80.209 (87.175.80.209)
- Transmission Control Protocol, Src Port: http (80), Dst Port: 65198 (65198), Seq: 165601, Ack: 580, Len: 658
- [Reassembled TCP segments (166258 bytes): #890(1440), #891(1440), #893(1440), #895(1440), #896(1440), #899(1440), #901(1440), #902(1440), #905(1440)]
- Hypertext Transfer Protocol
 - HTTP/1.1 200 OK\r\n
 - Date: Mon, 28 Jun 2010 11:26:21 GMT\r\n
 - Server: Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 PHP/5.2.6-1+lenny8 with Suhosin-Patch mod_ssl/2.2.9 openssl/0.9.8g\r\n
 - X-Powered-By: PHP/5.2.6-1+lenny8\r\n
 - Content-Disposition: attachment; filename="Blatt09.pdf"; size=165871;\r\n
 - Content-Length: 165871\r\n
 - Keep-Alive: timeout=15, max=100\r\n
 - Connection: Keep-Alive\r\n
 - Content-Type: application/pdf\r\n
 - \r\n
- Media Type

The status bar at the bottom indicates: Frame (frame), 720 bytes | Packets: 1239 Displayed: 7 Marked: 0 | Profile: Default



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 Benutzernamen 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 Hashfunktion
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)

Konsument: Druck Dienst (plaxo.com)

- Photos hochladen
 - Photos von flickr drucken
 - Holt sich Anfragetoken von flickr, und leitet Endnutzer zur verifikation zu flickr weiter
- Benutzer/Passwort eingeben und Anfrage von plaxo bestätigen, Endbenutzer zu plaxo weiterleiten
 - 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?