Netzwerkanalyse für Entwickler

Von Thomas Kestler

Einleitung

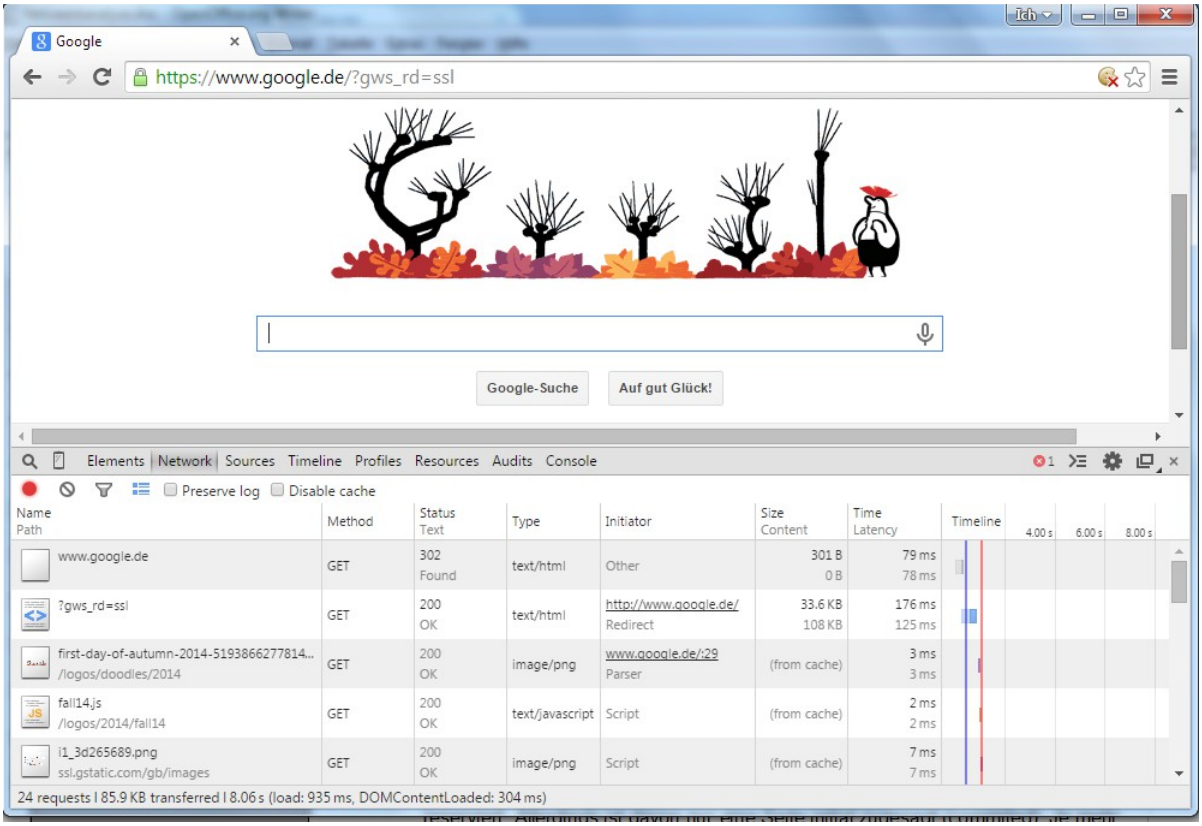
Ohne Netzwerk geht in der modernen IT nichts, ob LAN-Zugriffe zwischen Client und Server oder Server-Zugriffe auf externe Dienste. Während der Entwicklung besteht also ständig Bedarf an einer Analyse des Netzwerkverkehrs. Moderne Web-Anwendungen setzen AJAX, REST, usw. meist asynchron ein, so dass da schon viel Verkehr parallel über die Leitung geht. Für die Analyse von Fehlern oder Lastverhalten muss ein Entwickler also in der Lage sein qualifiziert zu analysieren.

Im Folgenden stelle ich einige Tools zur Netzwerkanalyse vor, die jeder Entwickler kennen sollte.

Browser Tools

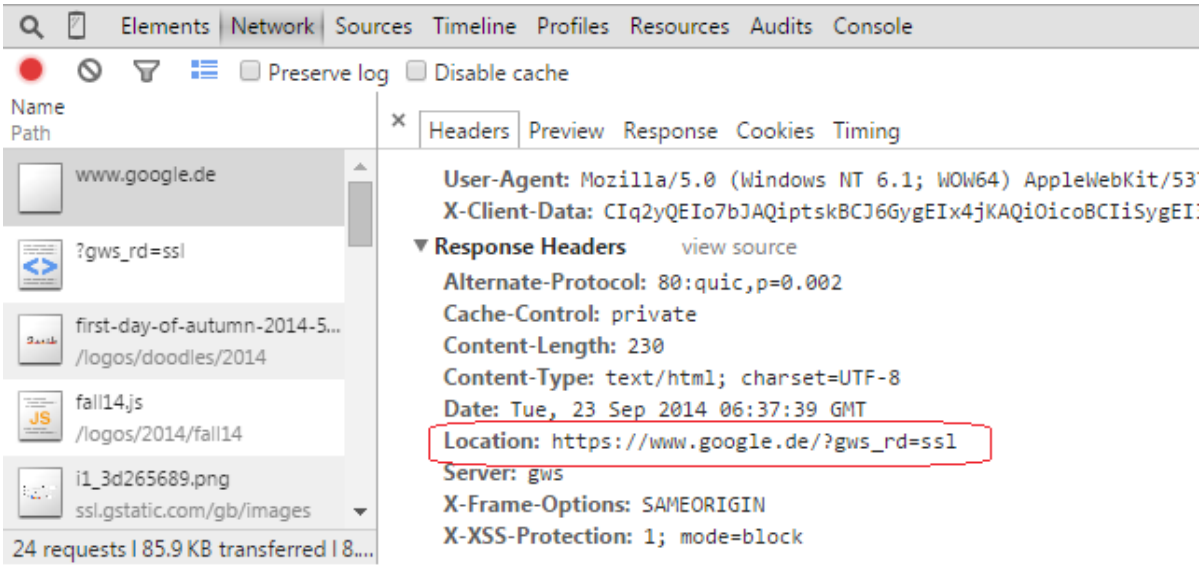
Alle modernen Browser bieten inzwischen integrierte Werkzeuge zur Analyse an. Firebug für Firefox war der Pionier, Google zog nach und auch Microsoft hat in mehreren Schritten aufgeholt. Allen gemeinsam ist der Aufruf mit der F12-Taste, alle erlauben Inspektion der Quellen (HTML, JS), Javascript-Debugging und Analyse des Netzwerkverkehrs.

Ohne solche Tools kann man moderne Web-Applikationen kaum mehr effizient entwickeln. Gerade der Javascript-Debugger ist heute wichtig, da viele Web-Anwendungen massiv Javascript einsetzen. Aber auch für die Netzwerkanalyse eignen sich solche Tools sehr gut:

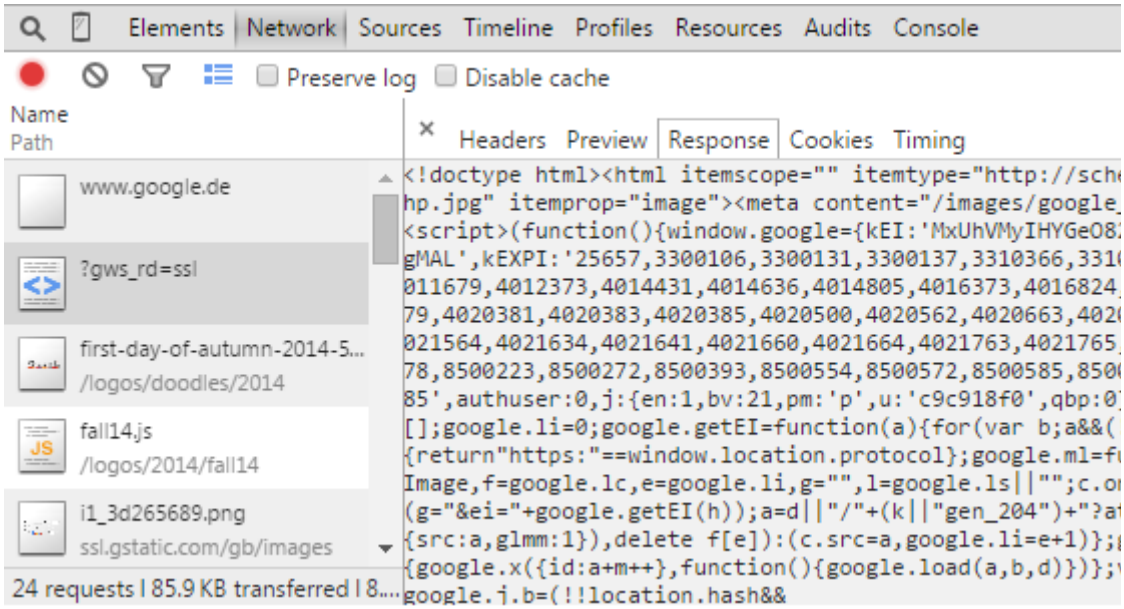


Eingegeben wurde zunächst google.de (aufgelöst von Chrome nach http://www.google.de) und dieser Request wurde an google.de abgesendet und mit einer Response 302 (Found, vormals Redirected Temporarily) auf die https-URL.

Alle Details der Anfrage und der Antwort (Response) kann man anzeigen lassen:

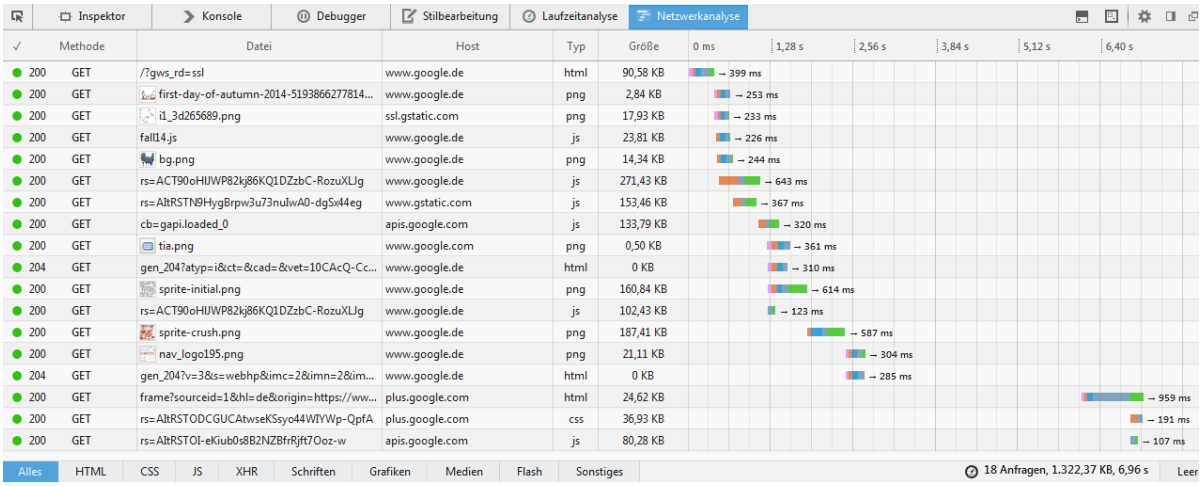


Der Browser folgt dieser Umleitung und greift nun auf https://www.google.de zu (dargestellt als ?gws_rd=ssl, dem sog. Path). Als Antwort erhält der Browser die HTML-Seite:

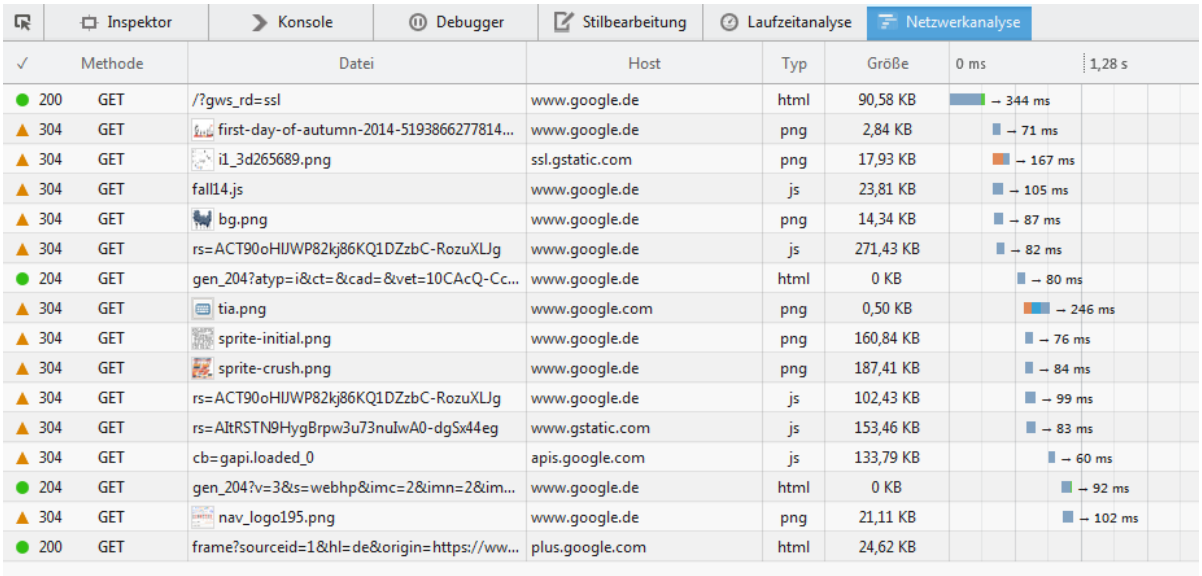


In der HTML-Seite werden nun weitere Ressourcen referenziert, so das herbstliche Google-Bild (first-day...), sowie weitere Javascript- und Image-Dateien. Aus dem ersten Screenshot oben ist ersichtlich nach welcher Zeit der DOM-Content (HTML-Seite) geladen und wann sie komplett geladen wurde (inkl. Javascript-Aktionen). Wir

können ersehen, dass der einfache Aufruf von google.de 24 Requests nach sich gezogen hat. Hierbei können wir noch mehr erfahren, nämlich, ob die Daten komplett übertragen wurden oder aus dem Browser-Cache entnommen wurden. Hier gibt es mehrere Mechanismen zum Caching: Zum einen kann der Web-Server ein Expires-Header in der Response mit senden, das dem Browser erlaubt, die Ressource eine Zeit lang zu cachen, zum anderen kann der Web-Server einen ETag-Header senden, den der Browser bei einer späteren Anfrage zu dieser Ressource vorlegt und der Web-Server kann entscheiden, ob die Ressource neu ausgeliefert werden muss oder vom Browser aus dem Cache genommen werden soll (http Statuscode 304 Not Modified). Leider schummelt Google Chrome hier etwas, weil ein Statuscode 200 simuliert wird, für Resourcen, die aus dem Cache kommen. Sehen wir uns die Sache mal im Firefox an:

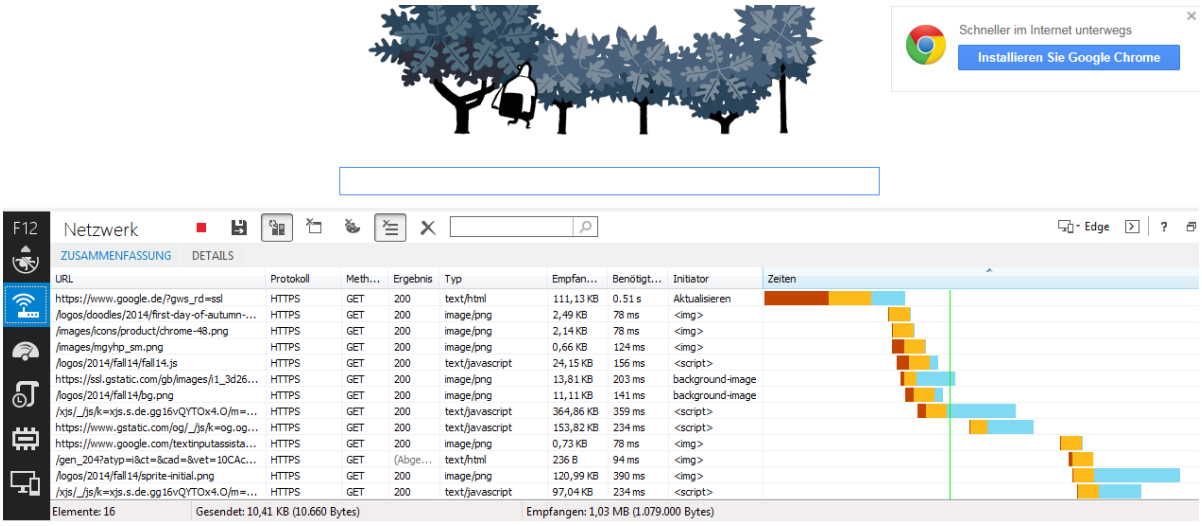


Hier das Protokoll eines „frischen“ Aufrufes, provoziert per Ctrl-F5, um den Cache komplett zu umgehen. Der Browser sendet nun keine ETags mit, der Web-Server muss alles ausliefern. Nun ein erneutes Laden mit F5:



Nun sehen wir viele 304 Antworten, da sich die Ressourcen nicht geändert haben und der Browser diese aus dem Cache verwenden darf.

Zum Schluss noch das gleiche im IE 11 (Reload per Ctrl-F5):



Niederschmetternd ist die Erkenntnis, dass die einst so schlanke Google-Startseite hier mit mehr als 1 MB zu Buche schlägt, aber immerhin ist auch das eine Erkenntnis.

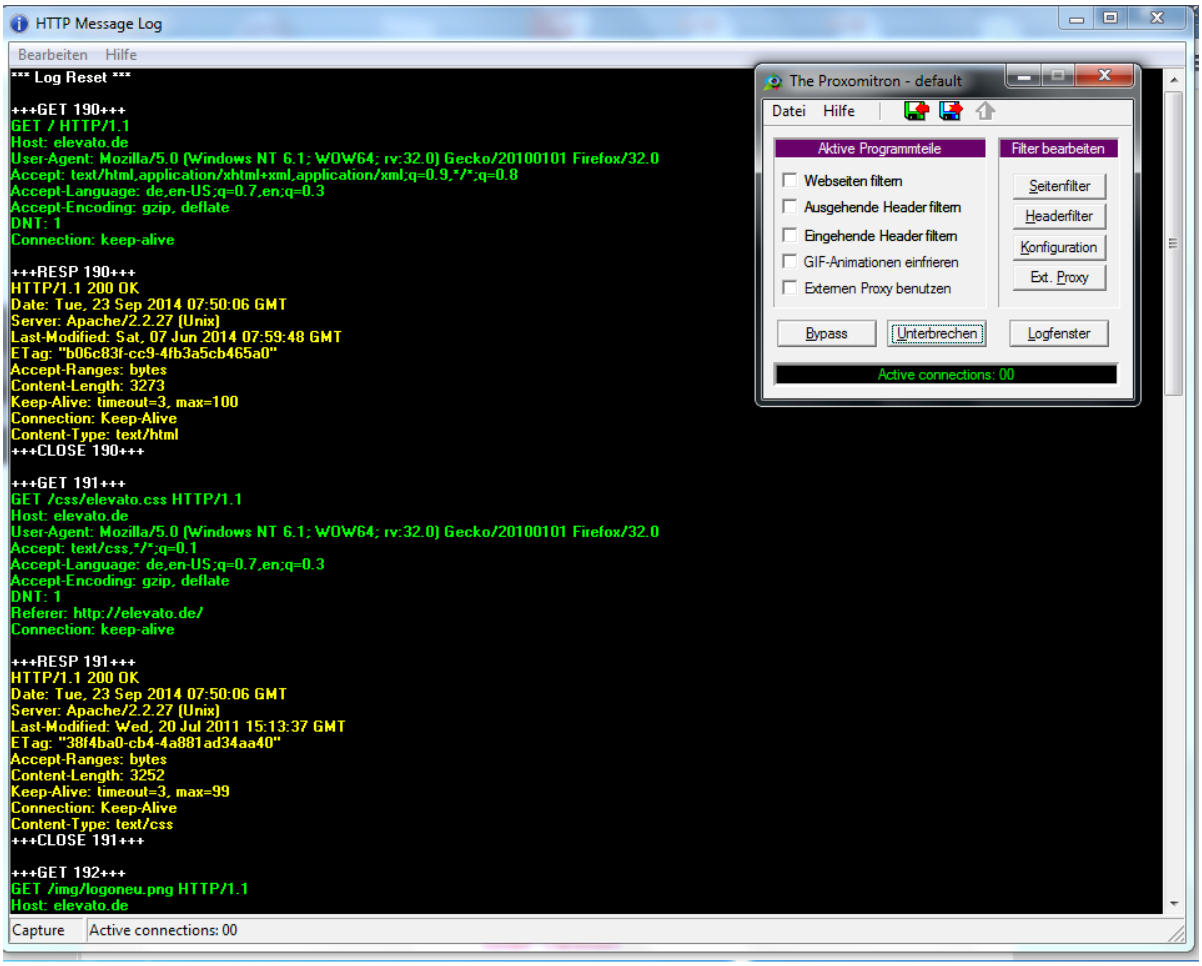
Wir konnten also sehen, dass alle modernen Browser (Opera fehlt hier, der hat mit DragonFly ein vergleichbares Tool) hilfreiche Tools zur Netzwerkanalyse mitbringen. Jeder Entwickler sollte sich damit vertraut machen.

Proxomitron

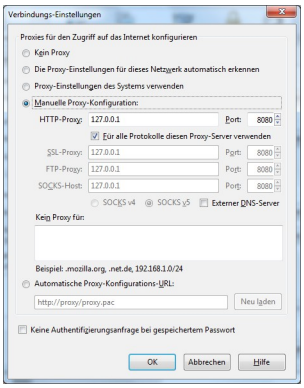
Ein weiteres, sehr hilfreiches Werkzeug ist Proxomitron, ein lokaler Proxy, mit Analyse- und Filterfunktionen. In Unternehmen erfolgt der Zugriff auf Intranet und Internet häufig über Web-Proxies (z. B: Squid) und diese verfälschen unter Umständen das Netzwerkverhalten erheblich. Auch in Mobilfunknetzen pfuschen einem ISP-Caches vehemment ins Handwerk, ich hatte das an anderer STelle ja bereits beschrieben[1].Weiterhin verhalten sich die Browser unterschiedlich, je nachdem ob sie sich direkt mit dem Internet verbunden glauben oder über einen Proxy (ob ein transparentes Port-Forwarding zu einem Web-Proxy erfolgt, kann der Browser nicht immer feststellen). So wird die Anzahl paralleler Connections bei Proxy-Betrieb deutlich reduziert, was AJAX-Anwendungen behindern kann (die lokal prima liefen, dann im Intranet aber nicht mehr).

Daher ist es sinnvoll, die Web-Anwendung als Entwickler über einen Proxy wie Proxomitron zu testen. Das Setup ist minimal: ZIP downloaden und entpacken, starten, konfigurieren, neu starten, Browser auf lokalen Proxy umstellen und los geht es.

1 Thomas Kestler, Mythos Proxy, Capalogic GmbH 2012



Das obige Bild zeigt einen laufenden Proxomitron (Port 8080) mit Logfenster und die Ausgaben für den Zugriff aus Firefox auf http://elevato.de. Der Firefox muss zuvor so eingestellt werden, dass er diesen Proxy benutzt:

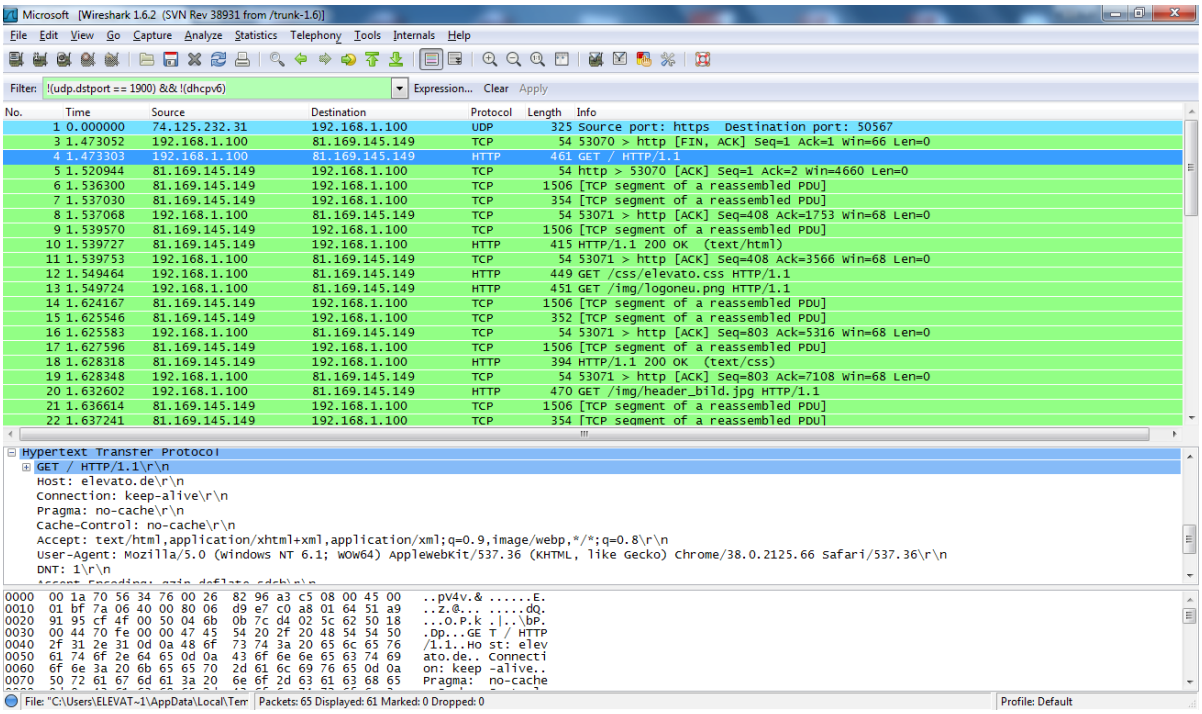


Wireshark

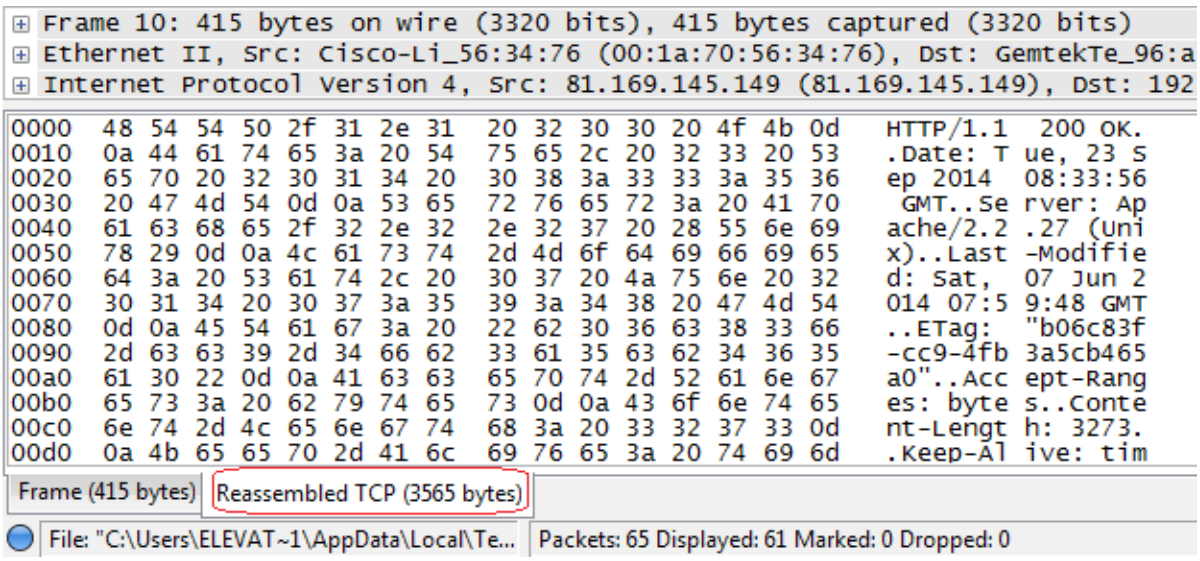
Der „Next Level“ der Netzwerkanalyse beginnt mit Wireshark, einem Tool zum Mitschneiden und analysieren von Netzwerkverkehr. Das ist schon anspruchsvoller und je nach dem an welcher Stelle man welchen Verkehr abgreift auch ggf. sensibel (Datenschutz, Datensicherheit). Wireshark ist kostenlos, benötigt zum Mitschneiden

unter Windows jedoch den PCAP Treiber (Linux: tcpdump), was aber bei der Installation automatisch erfolgt.

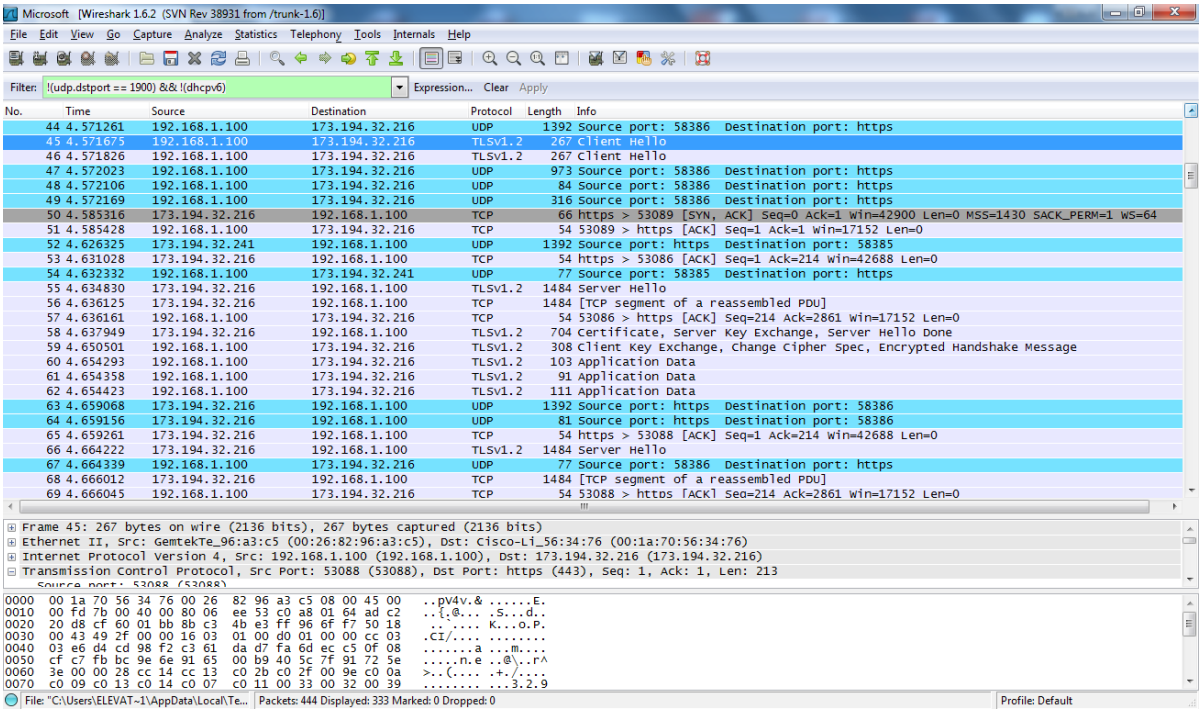
Wireshark ist mächtig, aber anfangs schwer zu verstehen, zumal da ja ziemlich viel über die LEitung läuft, was nicht interessiert, so z. B. SSDP Verkehr des UPnP-Protokolls. Damit und mit vielem anderen läuft einem erst mal das Logfenster zu, auch ohne Interaktion. Daher sollte man zunächst einen Filter setzen, um diese Geräusche auszublenden. Hier mal ein Mitschnitt eines Zugriffs mit Chrome auf http://www.elevato.de :



Man sieht den GET-Request auf elevato.de und die darauf folgende Antwort (Frames 5 – 9). Da die Anwort mehrere Ethernet-Pakete ausmacht, werden mehrere Frames erzeugt und Wireshark zeigt zunächst diese Frames als Teil der TCP-Kommunikation an. Erst im Frame 10 ist die Response komplett und nun kann Wireshark das ganze zu einer http-Response zusammenfassen:



Man sieht also schon, es ist kniffelig, den http Verkehr zu verfolgen. Noch komplexer wird es bei https, hier Google (wir erinnern uns, Google macht redirect auf https):

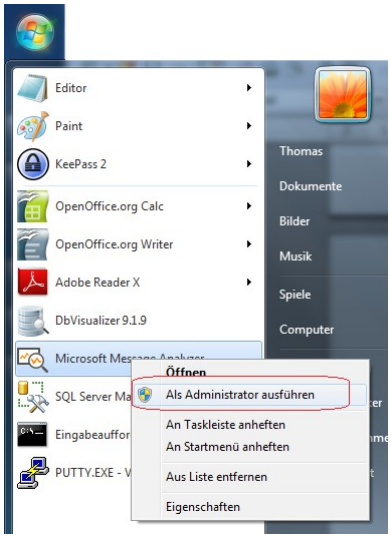


Man sieht den SSL-Handshake, die Daten des http-Verkehrs sind dann natürlich nicht mehr lesbar, da verschlüsselt. Wireshark kann SSL entschlüsselt, wenn man den Private Key des Servers hinterlegt. Den Private Key von Google haben wir natürlich nicht (und hoffentlich auch niemand sonst außer Google), aber bei unserem eigenen (Entwicklungs-)Server sollte das möglich sein. In den Weblinks befindet sich ein Link auf ein Blog, wie man SSL mit Wireshark entschlüsseln kann.

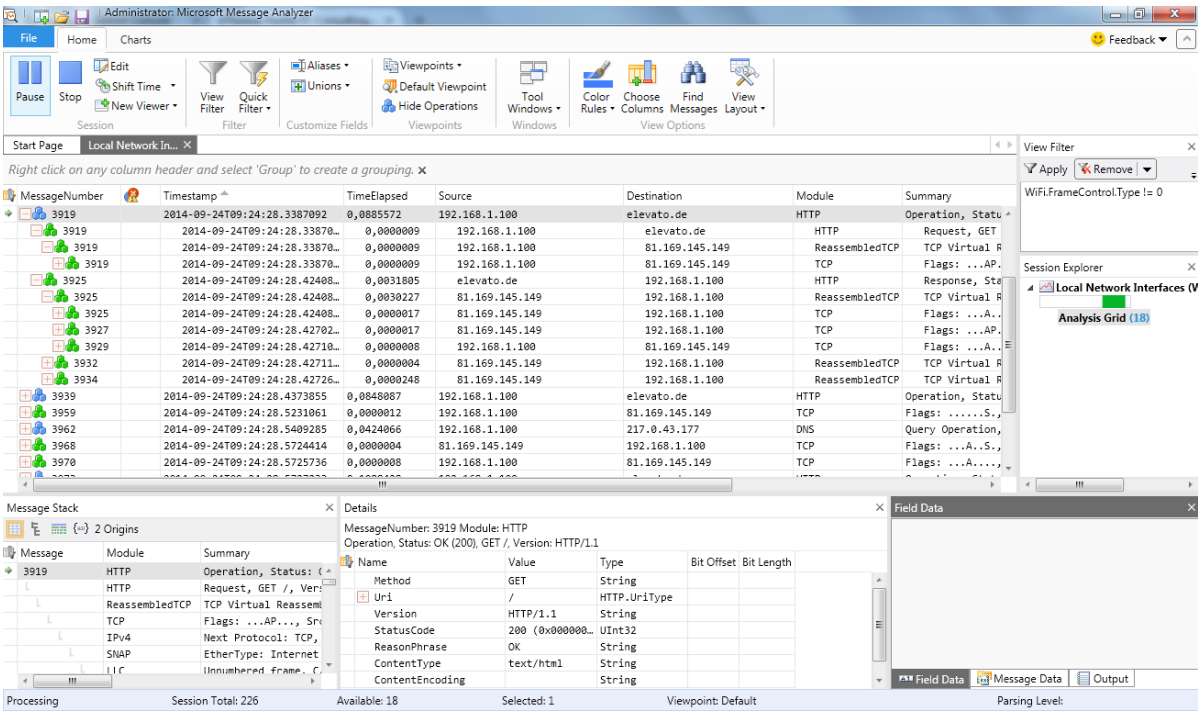
Message Analyzer

Mit Message Analyzer (MA) hat Microsoft ein Pendant zu Wireshark entwickelt, wobei MA noch weiter geht: Aufbauen auf dem Event Trace for Windows (ETW) können auch andere Event-Messages analysiert werden, z. B. USB-Datenverkehr. Damit ist MA durchaus mächtiger als Wireshark. Die inzwischen vorliegende Version 1.1 ist ausgereift und bedienfreundlicher als Wireshark. Die Installation ist einfach, will man SSL/https analysieren muss man allerdings noch Fiddlercore von Telerik installieren.

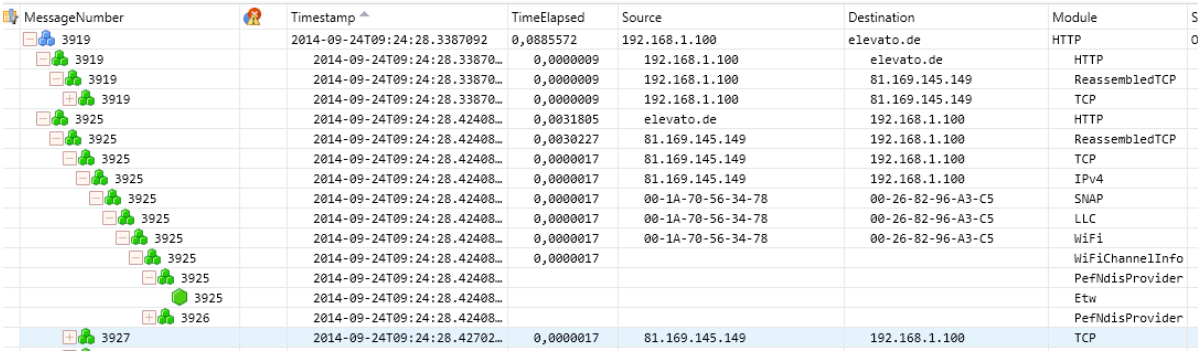
Der Start von MA muss mit Administratorrechten erfolgen, andernfalls ist kein Zugriff möglich (die Fehlermeldung dazu bringt einen nicht weiter).



Anschließend startet man eine neue Session (z. B. File->Quick Trace → Local...) und MA startet mit dem Mitschneiden des Netzwerkverkehrs von diesem Interface. Auch hier macht es Sinn erst mal Nebengeräusche weg zu filtern, dazu gibt es bereits eien vordefinierten Filter (Apply danach nicht vergessen). Im Folgenden ein Mitschnitt eines Zugriffes mit Firefox auf elevato.de (Ctrl-F5, um alles zu laden):

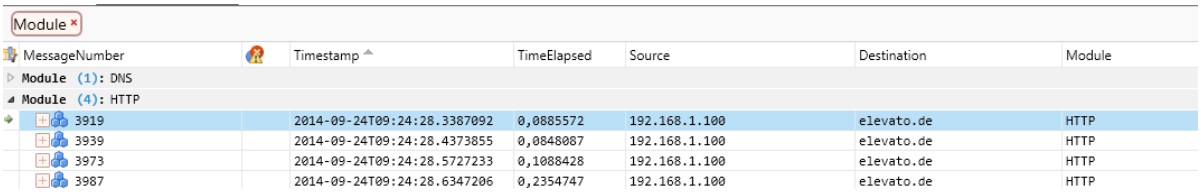


MA erkennt das höchstwertige Protokoll http und gruppiert die darunter liegenden Frames entsprechend ein. Wir sehen mit Frame 3919 den GET-Request und 3925 die (logische) http-Response, die in Wirklichkeit aus 6 TCP-Frames besteht. Man kann jeden TCP-Frame weiter aufklappen und man bekommt ein Gefühl für die Komplexität:

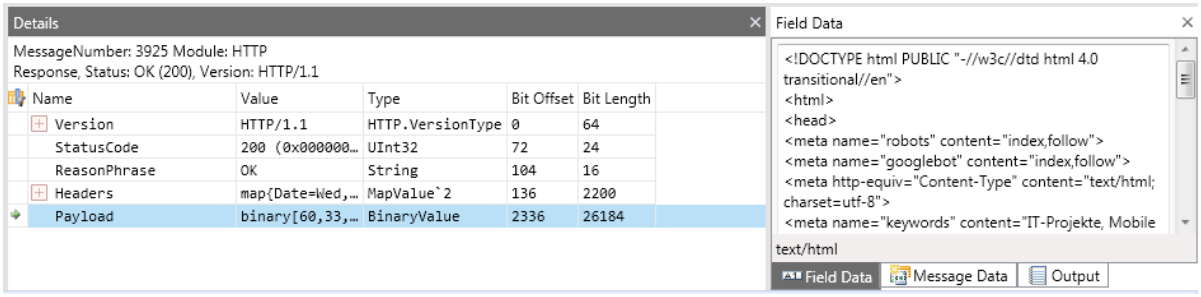


Der TCP-Frame besteht hier aus einem IP-Frame, SNAP[2] ist das Subnet Access Protokol, LLC der Logical Link, WiFI der Wireless-Stack, naja und so weiter. Man sieht also wie viele Events im ETW für einen einzigen Request erzeugt werden. Aber kaum ein Entwickler muss jemals so tief einsteigen. Normalerweise reicht die Analyse auf http-Level völlig aus und die ist in MA durchaus gut gelungen, insbesondere wenn man das Group-Feature nutzt (auf Column-Header rechte Maus → Group):

2 http://en.wikipedia.org/wiki/Ethernet_frame#IEEE_802.2_SNAP



So hat man den relevanten http-Verkehr auf einen Blick und kann eintauchen. Über die Details-View hat man Zugriff auf die Headers und Antwortdaten (HTML, JS, etc.):



Message Analyzer ist sehr mächtig und geht weit über Netzwerkanalyse hinaus. Es lohnt sich also auf jeden Fall sie das Tool anzusehen.

Zusammenfassung

Jeder Entwickler sollte sich mit den Möglichkeiten der Netzwerkanalyse vertraut machen, denn früher oder später wird eine Analyse fällig. Die vorgestellten Tools sind allesamt kostenlos und sehr hilfreich. Ob man lieber Wireshark oder MA nutzt, ist Sache des Geschmacks und des Betriebssystems (Wireshark ist auf vielen Plattformen wie LINUX, UNIX, Mac OSX, etc. verfügbar). Noch ein Hinweis: Die unterschiedlichen Browser legen ein stark unterschiedliches Netzwerkverhalten an den Tag (z. B. Concurrent Connections) – was in einem Browser gut läuft, muss in anderen noch lange nicht so problemlos laufen.

Weblinks

Proxomitron Homepage: http://www.buerschgens.de/Prox/

Firebug Homepage: http://getfirebug.com/

Wireshark Homepage: https://www.wireshark.org/

Blogbeitrag SSL mit Wireshark: http://www.foteviken.de/?p=2227

Download Message Analyzer: http://www.microsoft.com/en-us/download/details.aspx?id=40308

Message Analyzer Dokumentation: http://technet.microsoft.com/en-us/library/jj649776.aspx

Überblick über ETW: http://msdn.microsoft.com/en-us/magazine/cc163437.aspx

Fiddlercore von Telerik: http://www.telerik.com/fiddler/fiddlercore

Thomas Kestler ist Inhaber der Firma Thomas Kestler IT-Consulting (vormals elevato GmbH). http://www.elevato.de