Netzwerkanalyse für Entwickler

Von Thomas Kestler

Einleitung	2
Browser Tools	2
Proxomitron	5
Wireshark	6
Message Analyzer	
Zusammenfassung	
Weblinks	

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:

And and a second							l	Idh 🗸			
Soogle ×	Deaths 1ge	a began to	-								
← → C 🔒 https://www.google	.de/?gws_r	d=ssl							\$	ເ <u>ລີ</u> =	:
1	X	G	bogle-Suche	Auf gut Glück!		Â.					•
											-
Elements Network Sources Time	line Profiles	Resources A	udits Console					01	>= #	+ \$1⊡ x	
O Preserve log Disa	ble cache	Resources A						•-	/- 1	• ⊑⊿∩	-
Name Path	Method	Status Text	Туре	Initiator	Size Content	Time Latency	Timeline	4.00 s	6.00 s	8.00 s	
www.google.de	GET	302 Found	text/html	Other	301 B 0 B	79 ms 78 ms					
?gws_rd=ssl	GET	200 OK	text/html	http://www.google.de/ Redirect	33.6 KB 108 KB	176 ms 125 ms					1
first-day-of-autumn-2014-5193866277814 /logos/doodles/2014	GET	200 OK	image/png	www.google.de/:29 Parser	(from cache)	3 ms 3 ms					
fall14.js /logos/2014/fall14	GET	200 OK	text/javascript	Script	(from cache)	2 ms 2 ms					
i1_3d265689.png ssl.gstatic.com/gb/images	GET	200 OK	image/png	Script	(from cache)	7 ms 7 ms					•
24 requests I 85.9 KB transferred I 8.06 s (load: 9	35 ms, DOMCo	ntentLoaded: 3	04 ms)								

Eingegeben wurde zunächst google.de (aufgelöst von Chrome nach <u>http://www.google.de</u>) 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 <u>https://www.google.de</u> 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:

R	🗇 Inspektor	≫ K	Console	O Debugger	2 9	itilbearbeitung	🕑 La	ufzeitanaly	se 🔚 Netz	twerkanalyse							Þ	0,	#	0 0
~	Methode		Date	ei		Host		Тур	Größe	0 ms	1,2	28 s	2,56 s	3,84 s		5,12 s		6,40 9		
• 200	GET	/?gws_rd=ssl			www.go	ogle.de		html	90,58 KB	399	ms									
• 200	GET	🔛 first-day-	of-autumn-2	2014-5193866277814	www.go	ogle.de		png	2,84 KB	🛄 - 3	253 ms									
• 200	GET	i1_3d2656	89.png		ssl.gstati	c.com		png	17,93 KB	III → 2	233 ms									
• 200	GET	fall14.js			www.go	ogle.de		js	23,81 KB	🔲 - 2	226 ms									
• 200	GET	🙀 bg.png			www.go	ogle.de		png	14,34 KB	- 11	244 ms									
• 200	GET	rs=ACT90oHI	JWP82kj86K	Q1DZzbC-RozuXLJg	www.go	ogle.de		js	271,43 KB		- 64	3 ms								
• 200	GET	rs=AltRSTN9H	HygBrpw3u7	3nuIwA0-dgSx44eg	www.gst	atic.com		js	153,46 KB		→ 367	ms								
• 200	GET	cb=gapi.load	ed_0		apis.goo	gle.com		js	133,79 KB		-	- 320 ms	s							
• 200	GET	🔲 tia.png			www.go	ogle.com		png	0,50 KB			→ 361	ms							
• 204	GET	gen_204?atyp	=i&ct=&ca	d=&vet=10CAcQ-C	www.go	ogle.de		html	0 KB			→ 310 i	ms							
• 200	GET	sprite-init	ial.png		www.go	ogle.de		png	160,84 KB			-	614 ms							
• 200	GET	rs=ACT90oHI	JWP82kj86K	Q1DZzbC-RozuXLJg	www.go	ogle.de		js	102,43 KB		. .	123 ms								
• 200	GET	🔣 sprite-cru	sh.png		www.go	ogle.de		png	187,41 KB				→ 587 ms							
• 200	GET	🔚 nav_logo1	L95.png		www.go	ogle.de		png	21,11 KB				■ → 304 ms							
• 204	GET	gen_204?v=38	8ts=webhp8	kimc=28timn=28tim	www.go	ogle.de		html	0 KB				📕 → 285 ms							
• 200	GET	frame?source	id=1&hl=de	e&origin=https://wv	plus.god	gle.com		html	24,62 KB										→ 9	/59 ms
• 200	GET	rs=AltRSTOD	CGUCAtwse	KSsyo44WIYWp-Qpf	A plus.goo	gle.com		CSS	36,93 KB										- 1	91 ms
• 200	GET	rs=AltRSTOI-	eKiub0s8B2N	NZBfrRjft7Ooz-w	apis.goo	gle.com		js	80,28 KB										→ 10	/7 ms
Alles	HTML	CSS JS	XHR	Schriften	Grafiken	Medien	Flash	Sonsti	ges					Ø	18 An	fragen, 1.	322,37	KB, 6,9	96 s	Leer

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:

R	🖨 Inspektor	➤ Konsole	Debugger	Stilbearbeitung	🕑 La	ufzeitanalyse	🔄 Netz	werkanalyse	
\checkmark	Methode	Date	i	Host		Тур	Größe	0 ms	1,28 s
• 2	200 GET	/?gws_rd=ssl		www.google.de		html	90,58 KB	→ 344 ms	
A 3	304 GET	🔝 first-day-of-autumn-2	014-5193866277814	www.google.de		png	2,84 KB	■ → 71 ms	
	304 GET	i1_3d265689.png		ssl.gstatic.com		png	17,93 KB	📕 → 167 m	s
	304 GET	fall14.js		www.google.de		js	23,81 KB	■ → 105 ms	
	304 GET	🙀 bg.png		www.google.de		png	14,34 KB	■ → 87 ms	
A 3	304 GET	rs=ACT90oHIJWP82kj86K0	(1DZzbC-RozuXLJg	www.google.de		js	271,43 KB	■ → 82 ms	
•	204 GET	gen_204?atyp=i&ct=&cad	=&vet=10CAcQ-Cc	www.google.de		html	0 KB	I → 80 s	ns
A 3	304 GET	🥅 tia.png		www.google.com		png	0,50 KB	■ ■ →	246 ms
	304 GET	sprite-initial.png		www.google.de		png	160,84 KB	I → 76	ms
	304 GET	Exprite-crush.png		www.google.de		png	187,41 KB	■ → 84	ms
A 3	304 GET	rs=ACT90oHIJWP82kj86KC	1DZzbC-RozuXLJg	www.google.de		js	102,43 KB	■ → 99	ms
A 3	304 GET	rs=AItRSTN9HygBrpw3u73	8nuIwA0-dgSx44eg	www.gstatic.com		js	153,46 KB	■ → 83	ms
A 3	304 GET	cb=gapi.loaded_0		apis.google.com		js	133,79 KB	I -	60 ms
• 2	204 GET	gen_204?v=3&s=webhp&	imc=2&imn=2&im	www.google.de		html	0 KB		→ 92 ms
	304 GET	nav_logo195.png		www.google.de		png	21,11 KB		l → 102 ms
•	200 GET	frame?sourceid=1&hl=de	&origin=https://ww	plus.google.com		html	24,62 KB		

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

			-		Ŷ	, T	T			0	Schneller im Int	ernet unterw	egs Chrome	e
F12	Netzwerk 📕 👪	N X-	۵ (×		Q						⊊ <u>1</u> - Edge	Σ	? 8
(7335-)	ZUSAMMENFASSUNG DETAILS													
T	URL	Protokoll	Meth	Ergebnis	Тур	Empfan	Benötigt	Initiator	Zeiten		*			
\otimes	https://www.google.de/?gws_rd=ssl	HTTPS	GET	200	text/html	111,13 KB	0.51 s	Aktualisieren						
<u> </u>	/logos/doodles/2014/first-day-of-autumn	HTTPS	GET	200	image/png	2,49 KB	78 ms							
	/images/icons/product/chrome-48.png	HTTPS	GET	200	image/png	2, 14 KB	78 ms							
	/images/mgyhp_sm.png	HTTPS	GET	200	image/png	0,66 KB	124 ms							
	/logos/2014/fall14/fall14.js	HTTPS	GET	200	text/javascript	24, 15 KB	156 ms	<script></script>						

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 beschrieben¹.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 <u>http://elevato.de</u>. Der Firefox muss zuvor so eingestellt werden, dass er diesen Proxy benutzt:

Proxy-Einstellu	ngen des Systems verwenden		
Manuelle Proxy	-Konfiguration:		
HTTP-Proxy:	127.0.0.1	Port:	8060
	Eür alle Protokolle diesen Proxy	Server verw	enden
SSL-Praxy:	127.0.0.1	Port:	8060
FTP-Progy:	127.0.0.1	Port:	8080
SOCKS-Host:	127.0.0.1	Port:	8080
Beispiel: .mozil	la.org, .net.de, 192.168.1.0/24 Prov. Konfigurations. I IRI -		

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 <u>http://www.elevato.de</u> :

7	Aicrosoft (Wi	reshark 1	.6.2 (SVN Rev 38	931 from /trur	k-1.6)]					-			-		
Eile	Edit View	<u>G</u> o <u>C</u>	apture <u>A</u> nalyze	Statistics	Telephony <u>T</u> ools	Internals	<u>H</u> elp								
			😑 🗔 🗙 🗟	BIQ	- 	2 1		0, 🖭 🕯	K 🖂 🍢	6 🖬					
Filt	er: !(udp.dstp	ort == 19	900) && !(dhcpv6)		- Expre	ession Clear Aj	oply							
No.	Time		Source		Destination		Protocol	Length Info							
	1 0.00	0000	74.125.23	2.31	192.168.1.	100	UDP	325 50	ince port:	https	Destinati	on port: 505	67		
	3 1.47	3052	192.168.1.	.100	81.169.14	5.149	TCP	54 53)70 > http	[FIN,	ACK] Seq=1	. Ack=1 Win=6	6 Len=0		
	4 1.47	3303	192.168.1	.100	81.169.14	5.149	нттр	461 GE	г / НТТР/1	.1					_
	5 1.52	0944	81.169.14	5.149	192.168.1.	100	TCP	54 ht	p > 53070	[ACK]	Seq=1 Ack=	2 W1n=4660 L	en=0		
	6 1.53	6300	81.169.14	5.149	192.168.1.	100	TCP	1506 [TO	P segment	ofar	eassembled	I PDU]			
	/ 1.53 9 1 52	7030	81.169.14	100	192.168.1	100	TCP	354 LIC 54 52	P segment	or a r	eassembled	1 PDU] :k_1752 win_6	8 L on=0		
	0 1. 33	0000	81 160 14	5 140	102 168 1	100	TCP	1506 [T/	D sogmont	of a r	Seq=408 AC	.K=1/35 WIN=0	o Len=0		
	10 1 53	9727	81 169 14	5 149	192.168.1	100	нттр	415 HT	P 3egment	OK (†	ext/html)	PDOJ			
	11 1.53	9753	192,168,1	.100	81,169,14	5.149	TCP	54 53	71 > http	[ACK]	Seg=408 Ac	k=3566 win=6	8 Len=0		
	12 1.54	9464	192.168.1	100	81.169.14	5.149	HTTP	449 GE	/css/ele	vato.cs	s HTTP/1.1				
	13 1.54	9724	192.168.1	.100	81.169.14	5.149	HTTP	451 GE	/img/log	oneu.pn	ng HTTP/1.1				
	14 1.62	4167	81.169.14	5.149	192.168.1.	100	TCP	1506 [TO	P segment	ofar	eassembled	PDU]			
	15 1.62	5546	81.169.14	5.149	192.168.1.	100	TCP	352 [TO	P segment	ofar	eassembled	PDU]			
	16 1.62	5583	192.168.1.	.100	81.169.14	5.149	TCP	54 53)71 > http	[ACK]	Seq=803 Ac	:k=5316 Win=6	8 Len=0		
	17 1.62	7596	81.169.14	5.149	192.168.1.	100	TCP	1506 [TO	P segment	ofar	eassembled	PDU]			
	18 1.62	8318	81.169.14	5.149	192.168.1.	100	HTTP	394 HT	P/1.1 200	OK (t	ext/css)				
_	19 1.62	8348	192.168.1	.100	81.169.14	0.149	тср	54 530	0/1 > http	[ACK]	Seq=803 Ac	:k=/108 Win=6	8 Len=0		
	20 1.63	2602	192.168.1	100	81.169.14	100	HITP	4/0 GE	/img/nea	der_bii	a.jpg Hilf	//1.1			
	21 1.05	7241	81 160 14	5 149	102 168 1	100	TCP	254 [T	P segment	ofar	eassembled				
4	22 1.05	/ 241	01.109.14	J. 149	192.100.1.	100	ICF	554 11	.F Sequenc	UI a I	eassendred	FDOT			•
	whentext	Transt	er Protocol												
	БСТ / НТ	TP/1.1	\r\n												^
	Host: el	evato.	de\r\n												
	Connecti	on: ke	ep-alive\r\	n											
	Pragma:	no-cac	he\r\n												
	Cache-Co	ntrol:	no-cache\r	\n											
	Accept:	text/h	tml,applica	tion/xhtml	+xml,applica	tion/xml	;q=0.9,image	/webp,*/*	q=0.8\r\r						E
	User-Age	nt: Mo	zilla/5.0 (Windows NT	6.1; WOW64)	Applewe	ebKit/537.36	(KHTML, 1	ike Gecko)	Chrome	2/38.0.2125	5.66 Safari/5	37.36\r\n		
	DNT: 1\r	\n			-										-
000	0 00 1 -	70 56	34 76 00 26	82 96 23	c5 08 00 45	00	nV4v &	F.							
001	0 01 bf	7a 06	40 00 80 06	d9 e7 c0	a8 01 64 51	a9	z.@	iq.							
002	0 91 95	f 4f	00 50 04 6b	0b 7c d4	02 5c 62 50	18	.0.P.k . \l	DP.							=
003	0 00 44 1 0 2f 31	0 TE	00 00 4/ 45 0d 0a 48 6f	54 20 2t 73 74 3a	20 48 54 54 20 65 66 65	50 .D	рGE Т / Н .1. но st • •	lev							
005	0 61 74	of 2e	64 65 0d 0a	43 6f 6e	6e 65 63 74	69 at	o.de Conned	ti							
006	0 6f 6e	3a 20	6b 65 65 70	2d 61 6c	69 76 65 0d	0a on	: keep -alive	2							
007	0 50 /2 0	<u>1 6/</u>	60 61 3a 20	be of 2d	03 01 03 68	po Pr	agma: no-cao	-ne							Ŧ
	File: "C:\Users\	ELEVAT~	1\AppData\Loca	NTem Packet	s: 65 Displayed: 61	Marked: 0 D	ropped: 0							Profile: Default	

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:

+	Frame	10	: 41	5 b	yte	s oi	n w	ire	(33)	20	bit	s),	41	5 b	yte	s ca	aptu	red	(33	20	bits)	
+ [Etheri	net	II,	Sr	c: (cis	co-l	Li_!	56:34	4:7	6 (00:	1a:	70:	56:	34:7	76),	Dst	:: G	emt	ekTe_	96:a
+ :	Interi	net	Pro	toc	01	ver:	sio	n 4	, Sr(с:	81.3	169	.14	5.1	49	(81.	169	.14	5.14	9),	Dst:	192
000	00 48	3 54	54	50	2f	31	2e	31	20	32	30	30	20	4f	4b	0d	н	TTP/	1.1	2	00 ок	
001	LO 0a	a 44	61	74	65	3a	20	54	75	65	2c	20	32	33	20	53		Date	: т	ue	, 23	s
002	20 65	5 70	20 (32	30	31	34	20	30	38	Зa	33	33	Зa	35	36	e	p 20)14	08	:33:5	6
003	30 20) 47	′4d	54	0d	0a	53	65	72	76	65	72	Зa	20	41	70		GMT.	.se	rv	er: A	p
004	40 61	63	68	65	2f	32	2e	32	2e	32	37	20	28	55	6e	69	a	che/	2.2	. 2	7 (Un	i –
005	50 78	3 29) Od	0a	4c	61	73	74	2d	4d	6f	64	69	66	69	65	X)L	.ast	-M	odifi	e
000	50 64	3a	a 20	53	61	74	2c	20	30	37	20	4a	75	6e	20	32	d	: Sa	ıt,	07	Jun	2
007	70 30) 31	. 34	20	30	37	Зa	35	39	3a	34	38	20	47	4d	54	0	14 ()7:5	9:4	48 GM	Т
008	80 00	1 Oa	a 45	54	61	67	Зa	20	22	62	30	36	63	38	33	66		.ETa	ig:	"b	06c83	f
009	90 20	63	63	39	2d	34	66	62	33	61	35	63	62	34	36	35	-	cc9-	4fb	3a	5cb46	5
003	a0 61	. 30	22 (0d	0a	41	63	63	65	70	74	2d	52	61	6e	67	a	0".,	Acc	ep:	t-Ran	g
00	0 65	5 73	3 a	20	62	79	74	65	73	0d	0a	43	6f	6e	74	65	e	s: b	yte	s.	.Cont	e
000	CO 60	2 74	2 d	4c	65	6e	67	74	68	3a	20	33	32	37	33	0d	n	t-Le	engt	h:	3273	•
000	d0 0a	14k	65	65	70	2d	41	6C	69	76	65	3a	20	74	69	6d		Keep)-A I	iv	e: ti	m
Fra	me (415	byte	es) R	eass	embl	ed T	CP (3	3565	bytes)]												
\bigcirc	File: "C:	\Use	rs\ELE	VAT	~1\A	ppD	ata\l	.ocal	\Te	Pa	ckets	: 65 I	Displ	ayed	: 61 I	Mark	ed: 0	Dropp	oed: 0			

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):

М 🏹	icrosoft [Wiresh	ark 1.6.2 (SVN Rev 38931 f	from /trunk-1.6)]						x
<u>F</u> ile	Edit View G	o <u>C</u> apture <u>A</u> nalyze <u>S</u> ta	atistics Telephony <u>T</u> ools <u>I</u> r	ternals <u>H</u> elp					
	iii () () ()	(🖻 🗖 🗙 😂 🗄] 🔍 🗢 🛸 🎝 🚡 🛓	: 🔳 🖬 🗨 Q	0	🏽 🗹 🍢 💥 💢			
Filter	: !(udp.dstport :	== 1900) && (dhcpv6)		Expression Clear	Apply				
No.	Time	Source	Destination	Protocol	Length Ir	nfo			-
	44 4.57126	192.168.1.100	173.194.32.2	16 UDP	1392 5	Source port: 58386	Destination port: https		
	45 4.57167	'5 192.168.1.100) 173.194.32.2	16 TLSV1.2	267 C	lient Hello			
	46 4.57182	192.168.1.100	173.194.32.2	16 TLSV1.2	267 0	lient Hello	Barting and here		
	4/ 4.5/202	102.108.1.100	1/3.194.32.2	16 UDP	9/3 5	Source port: 58386	Destination port: https		E
	40 4. 37210	192.108.1.100	173.194.32.2	16 UDP	216 5	Source port: 58386	Destination port: https		
	50 4 58531	6 173 194 32 21	6 192 168 1 10	0 TCP	66 b	$t \pm ns > 53089$ [SYN	ACK] Seq=0 Ack=1 Win=42900 Len=0 MSS	5=1430 SACK PERM=1 WS=64	
	51 4, 58542	8 192,168,1,100	173,194,32,2	16 TCP	54 5	3089 > https [ACK]	Seg=1 Ack=1 Win=17152 Len=0		
	52 4.62632	5 173.194.32.24	1 192.168.1.10	0 UDP	1392 5	Source port: https	Destination port: 58385		
	53 4.63102	8 173.194.32.21	.6 192.168.1.10	0 TCP	54 h	nttps > 53086 [ACK]	Seg=1 Ack=214 Win=42688 Len=0		
	54 4.63233	192.168.1.100	173.194.32.2	41 UDP	77 5	Source port: 58385	Destination port: https		
	55 4.63483	173.194.32.21	.6 192.168.1.10	0 TLSv1.2	1484 5	Server Hello			
	56 4.63612	173.194.32.21	.6 192.168.1.10	0 ТСР	1484 [TCP segment of a r	eassembled PDU]		
	57 4.63616	192.168.1.100) 173.194.32.2	16 TCP	54 5	53086 > https [ACK]	Seq=214 Ack=2861 Win=17152 Len=0		
	58 4.63794	9 173.194.32.21	.6 192.168.1.10	0 TLSv1.2	704 C	ertificate, Server	Key Exchange, Server Hello Done		
	59 4.65050	1 192.168.1.100	173.194.32.2	16 TLSv1.2	308 C	lient Key Exchange	, Change Cipher Spec, Encrypted Hands	shake Message	
	60 4.65429	192.168.1.100	173.194.32.2	16 TLSv1.2	103 A	Application Data			
	61 4.65435	8 192.168.1.100	1/3.194.32.2	16 TLSV1.2	91 A	Application Data			
	62 4.65442	192.168.1.100	1/3.194.32.2	16 TLSV1.2	111 A	Application Data	Destination water 50200		
	63 4.65906	08 1/3.194.32.21	.6 192.168.1.10	0 000	1392 5	Source port: https	Destination port: 58386		
	65 4 65036	1 172 104 22 21	6 102 168 1 10		54 5	source port: https	Sog-1 Ack-214 wip-42688 Lop-0		
	66 4 66423	172 104 22 21	6 102 168 1 10	0 TLEV1 2	1494 0	Corver Hello	Seq=1 ACK=214 WITI=42088 Lefi=0		
	67 4 66433	19 192 168 1 100	173 194 32 2	16 100	77 9	Source port: 58386	Destination port: https		
	68 4 66601	2 173 194 32 21	6 192 168 1 10	0 TCP	1484 [TCP segment of a r	eassembled PDUl		
	69 4,66604	5 192,168,1,100	173,194,32,2	16 TCP	54 5	3088 > https [ACK]	Seg=214 Ack=2861 Win=17152 Len=0		-
4						11			F.
I F	ame 45: 267	bytes on wire (21	136 bits), 267 bytes (aptured (2136 bi	ts)				
ET ET	hernet II.	Src: GemtekTe 96:a	a3:c5 (00:26:82:96:a3:	c5). Dst: Cisco-	Li 56:34:	:76 (00:1a:70:56:34	:76)		- â
I II	ternet Prot	ocol Version 4. Sr	c: 192.168.1.100 (192	.168.1.100). Dst	: 173.194	4.32.216 (173.194.3	2,216)		
🗉 Tr	ansmission	Control Protocol,	Src Port: 53088 (5308	88), Dst Port: ht	tps (443)), Seg: 1, Ack: 1,	Len: 213		
	Source nort	· 52088 (52088)							*
0000	00 1a 70	56 34 76 00 26 82	96 a3 c5 08 00 45 00	pv4v.&	.E.				~
0010	00 fd 7b	00 40 00 80 06 ee	53 c0 a8 01 64 ad c2	{.@	d				
0020	00 43 49	2f 00 00 16 03 01	00 d0 01 00 00 cc 03		J. P.				=
0040	03 e6 d4	cd 98 f2 c3 61 da	d7 fa 6d ec c5 0f 08	am.					
0050	cf c7 fb	bc 9e 6e 91 65 00	b9 40 5c 7f 91 72 5e	n.e@\.	-r^				
0070	c0 09 c0	20 CC 14 CC 13 CU 13 CO 14 CO 07 CO	11 00 33 00 32 00 39	>(2.9				-
⊖ Fi	le: "C:\Users\ELE	/AT~1\AppData\Local\Te	Packets: 444 Displayed: 333 M	arked: 0 Dropped: 0	-			Profile: Default	

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 \rightarrow 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):

R 🗔 📂 🔒	Administra	tor: Microsoft Message Analyzer	-						
File Home	Charts								🙂 Feedback 🔻 🔨
Pause Stop	Edit Shift Tim New Viev	e • View Quick Filter Filter Customiz	rs ▼ Im Viewp ns ▼ Im Defau a Fields View Viewp	opints • It Viewpoint Operations wpoints	olor Choose ules • Columns M View Op	Find View essages Layout • tions			
Start Page Lo	cal Network	: In ×						<.>	View Filter ×
Right click on ar	ny column	header and select 'Group' to crea	ite a grouping. 🗙					· · ·	🐨 Apply 🗽 Remove 🖃 📮
🚯 MessageNumbe	er 👧	Timestamp A	TimeElapsed	Source	Destination		Module	Summary	WiFi.FrameControl.Type != 0
FA 3919		2014-09-24709:24:28.3387092	0.0885572	192.168.1.100	elevato.d	e	HTTP	Operation, Statur	
- 3919		2014-09-24T09:24:28.33870		192.168.1.100	elevat	- o.de	HTTP	Request. GET	
		2014-09-24709:24:28.33870		192.168.1.100	81,169	.145.149	ReassembledTCP	TCP Virtual B	
F - 39	19	2014-09-24709:24:28.33870	0.0000009	192.168.1.100	81,169	.145.149	TCP	Flags: AP	Caralian European V
3925		2014-09-24709:24:28.42408		elevato.de	192.16	8.1.100	HTTP	Response, Sta	Session Explorer
3925		2014-09-24T09:24:28.42408		81.169.145.149	192.16	8.1.100	ReassembledTCP	TCP Virtual R	Local Network Interfaces (V
+ 4 39	25	2014-09-24T09:24:28.42408		81.169.145.149	192.16	8.1.100	TCP	Flags:A	Archivia Crid (19)
H 39	27	2014-09-24T09:24:28.42702		81.169.145.149	192.16	8.1.100	TCP	Flags: AP.	Analysis Grid (16)
+ 🔒 39	29	2014-09-24T09:24:28.42710		192.168.1.100	81.169	.145.149	TCP	Flags:A =	
+ 🔒 3932		2014-09-24709:24:28.42711	0.0000004	81.169.145.149	192.16	8.1.100	ReassembledTCP	TCP Virtual R	
HA 3934		2014-09-24T09:24:28.42726		81,169,145,149	192.16	8.1.100	ReassembledTCP	TCP Virtual R	
+ 3939		2014-09-24T09:24:28,4373855	0,0848087	192.168.1.100	elevato.d	e	HTTP	Operation. Statu	
+ 🔒 3959		2014-09-24T09:24:28.5231061	0,0000012	192.168.1.100	81.169.14	5.149	TCP	Flags:S.,	
+ 3962		2014-09-24T09:24:28.5409285	0,0424066	192.168.1.100	217.0.43.	177	DNS	Query Operation,	
+ 🔒 3968		2014-09-24T09:24:28.5724414	0,0000004	81.169.145.149	192.168.1	.100	TCP	Flags:AS.,	
HA 3970		2014-09-24T09:24:28.5725736	0,0000008	192.168.1.100	81.169.14	5.149	TCP	Flags:A,	
				400 470 4 400				a the second	4 111
•								,	
Message Stack		×	Details				× Fi	eld Data	×
📃 ቼ 🛲 🖽 :	2 Origins		MessageNumber	: 3919 Module: HTTP :: OK (200), GFT / Version: HTTP/1	1				
🕼 Message	Module	Summary	B. Mana	Value	T	Dis Offices Dis Lowest			
3919	HTTP	Operation, Status: (*	ug Name	Value	туре	bit Offset bit Lengt	1		
	HTTP	Request, GET /, Ver:	Method	GET	String				
	Reassemb	ledTCP TCP Virtual Reassem	H Uri	/	HIIP.Urilype				
	TCP	Flags:AP, Sre	Version	HTTP/1.1	String		=		
	IPv4	Next Protocol: TCP,	Statuscoo	200 (0x000000.	UINC52				
	SNAP	EtherType: Internet	KeasonPhr	rase UK	String				
	LIC .	Unnumbered frame. C.*	Contently	/pe text/html	String			Marr	rago Data 🗐 Output
<		►	ContentEr	ICOUTUR	SURING		· ·	Held Data	age Data 🔚 Output
Processing		Session Total: 226	Available: 18	Selected: 1	Vie	wpoint: Default		Pars	ing Level:

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:

🚯 MessageNumber	Timestamp 🔷	TimeElapsed	Source	Destination	Module	S
- 🖧 3919	2014-09-24T09:24:28.3387092	0,0885572	192.168.1.100	elevato.de	HTTP	C
- 🔒 3919	2014-09-24T09:24:28.33870	0,0000009	192.168.1.100	elevato.de	HTTP	
- 🔒 3919	2014-09-24T09:24:28.33870	0,0000009	192.168.1.100	81.169.145.149	ReassembledTCP	
🖽 🔒 3919	2014-09-24T09:24:28.33870	0,000009	192.168.1.100	81.169.145.149	TCP	
- 🔒 3925	2014-09-24T09:24:28.42408	0,0031805	elevato.de	192.168.1.100	HTTP	
- 3925	2014-09-24T09:24:28.42408	0,0030227	81.169.145.149	192.168.1.100	ReassembledTCP	
- 3925	2014-09-24T09:24:28.42408	0,0000017	81.169.145.149	192.168.1.100	TCP	
- 者 3925	2014-09-24T09:24:28.42408	0,0000017	81.169.145.149	192.168.1.100	IPv4	
- 🖧 3925	2014-09-24T09:24:28.42408	0,0000017	00-1A-70-56-34-78	00-26-82-96-A3-C5	SNAP	
□♣ 3925	2014-09-24T09:24:28.42408	0,0000017	00-1A-70-56-34-78	00-26-82-96-A3-C5	LLC	
- 🔒 3925	2014-09-24T09:24:28.42408	0,0000017	00-1A-70-56-34-78	00-26-82-96-A3-C5	WiFi	
- 🔒 3925	2014-09-24T09:24:28.42408	0,0000017			WiFiChannelInfo	
- 🔒 3925	2014-09-24T09:24:28.42408				PefNdisProvider	
3925	2014-09-24T09:24:28.42408				Etw	
H 🔒 3926	2014-09-24T09:24:28.42408				PefNdisProvider	
Hanger 3927	2014-09-24T09:24:28.42702	0,0000017	81.169.145.149	192.168.1.100	TCP	

Der TCP-Frame besteht hier aus einem IP-Frame, SNAP² 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 \rightarrow Group):

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

N	lodule ×					
1	MessageNumber	Timestamp 📥	TimeElapsed	Source	Destination	Module
\triangleright	Module (1): DNS					
4 1	Module (4): HTTP					
+	H 🕂 🕂 3919	2014-09-24T09:24:28.3387092	0,0885572	192.168.1.100	elevato.de	HTTP
	H 🕂 🕂 3939	2014-09-24T09:24:28.4373855	0,0848087	192.168.1.100	elevato.de	HTTP
	H 🕂 🕂 3973	2014-09-24T09:24:28.5727233	0,1088428	192.168.1.100	elevato.de	HTTP
	H 🕂 🕂 3987	2014-09-24T09:24:28.6347206	0,2354747	192.168.1.100	elevato.de	HTTP

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.):

Details X						Field Data	×
MessageNumber: 3925 Module: HTTP Response, Status: OK (200), Version: HTTP/1.1					html PUBLIC "-//w3c//dtd html 4.0<br transitional//en">	*	
d)	Name	Value	Туре	Bit Offset	Bit Length	<html></html>	
	+ Version	HTTP/1.1	HTTP.VersionType	0	64	<head></head>	
	StatusCode	200 (0x000000	UInt32	72	24	<meta content="index,follow" name="robots"/>	
	ReasonPhrase	OK	String	104	16	<meta content="index,follow" name="googlebot"/>	
	+ Headers	<pre>map{Date=Wed,</pre>	MapValue`2	136	2200	charset=utf-8">	urrii;
4	Payload	binary[60,33,…	BinaryValue	2336	26184	<meta name="keywords" content="IT-Projekte, Mobile</td> <td>÷</td>	÷
						text/html	
						🏧 Field Data 🛛 🔠 Message Data 🖉 Output	

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: <u>http://www.microsoft.com/en-us/download/details.aspx?id=40308</u>

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

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). <u>http://www.elevato.de</u>