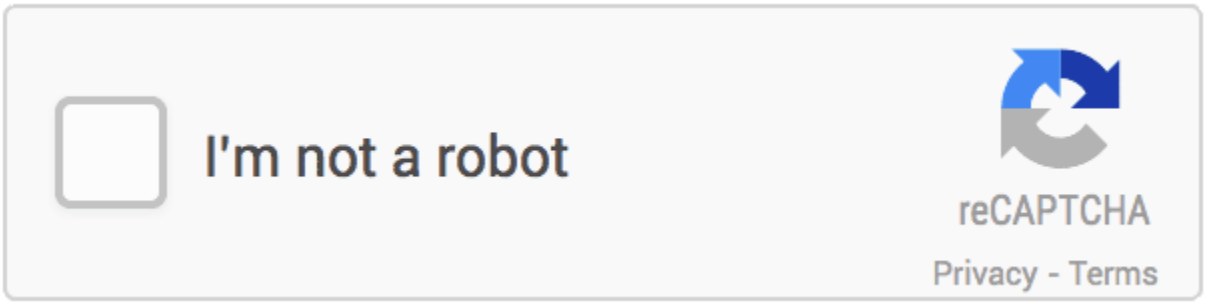
What's Happening Behind the Scenes of the Google reCAPTCHA Control

I implemented reCAPTCHA version 2 on my website, replacing the version 1 code I'd had in place since 2010. (Doing this on a .NET website was not as straightforward as the Google documentation might lead one to believe, by the way.)

The version 1 code made use of a DLL and registration of a web user control to present the user with a pair of words, or images with text, which the user must read and type.

The version 2 code dispenses with those in favor of a Web services-based approach to present the user with a checkbox control beside the prompt, "I'm not a robot", and if needed, a grid of images from which the user must make selections based on instructions ("Select all of the pictures which contain grass.")



I was seeing some long delays in both development and production environments when the contact form would load, and then the operation would time out in the production environment (but succeed in development). At length, I really became curious about what was going on during that page load in particular… so I fired up Wireshark to peek.

Wireshark

Wireshark is a software tool that allows the user to monitor network traffic. It records entries in a tabular format based on packet data and metadata: sent from, sent to, protocol type, payload (assuming the traffic isn't encrypted). There are books and other media available that discuss the ins-and-outs of using Wireshark to perform packet analysis.

Because my production site operates using a secure protocol, the information coming from monitoring activity there isn't quite as useful as the dev environment data, insofar as the data payloads are all encrypted (p.s. – I can confirm the secure protocols are working!).

Loading the Contact Form

I used Wireshark to record the events that followed clicking the "contact" link on a local development instance of my website. I wanted to look at the development instance partly because it uses plain ol' Hypertext Transfer Protocol – that is, no encryption.

The first few lines of code on the Web Form are used for loading various scripts and style sheets, most of which are located in local subdirectories, with one notable exception being the font service I use. The font service code is embedded in the master page, so it will always be the first code that reaches out from the server. The reCAPTCHA API code is the second such call – it dials Google to retrieve the JavaScript API.

```
<!--start recaptcha v2 API call-->
<script async defer src="https://www.google.com/recaptcha/api.js"></script>
<!--end recaptcha v2 API call-->
```

Notice please that the script is being called over an encrypted channel.

Summary

Wireshark shows that data is transacted with three servers: **www.google.com**, **www.gstatic.com**, and **gstaticadssl.l.google.com**. The server at **www.google.com** seems to broker the subsequent conversations, sort of like **www.google.com** was leading a telephone conference call. All of the conversations were completed in just over three and a half seconds.

Details

Let's look at these in parts generally corresponding to each conversation.

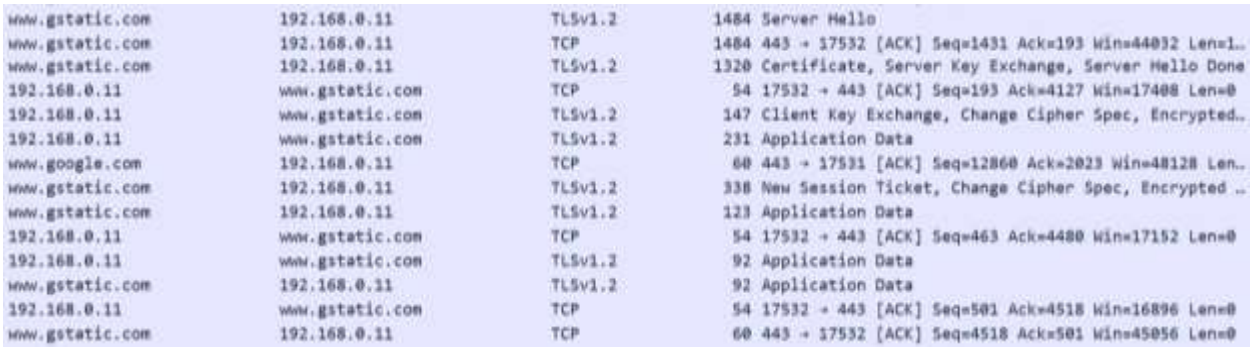
The first traffic is sent to **www.google.com** (recall the URL for the API we need is *https://www.google.com/recaptcha/api.js*). The conversation spanned 0.32s:

Source	Destination	Protocol	Length	Info
192.168.0.11	www.google.com	TCP	66	17531 → 443 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 W…
www.google.com	192.168.0.11	TCP	68	443 → 17531 [SYN, ACK] Seq=0 Ack=1 Win=42780 Len=0…
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1 Ack=1 Win=17408 Len=0
192.168.0.11	www.google.com	TLSv1.2	571	Client Hello
www.google.com	192.168.0.11	TCP	56	443 → 17531 [ACK] Seq=1 Ack=518 Win=44032 Len=0
www.google.com	192.168.0.11	TLSv1.2	210	Server Hello, Change Cipher Spec, Encrypted Handsh…
192.168.0.11	www.google.com	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
192.168.0.11	www.google.com	TLSv1.2	231	Application Data
192.168.0.11	www.google.com	TLSv1.2	836	Application Data
www.google.com	192.168.0.11	TCP	56	443 → 17531 [ACK] Seq=157 Ack=746 Win=45056 Len=0
www.google.com	192.168.0.11	TLSv1.2	123	Application Data
192.168.0.11	www.google.com	TLSv1.2	92	Application Data
www.google.com	192.168.0.11	TLSv1.2	92	Application Data
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1566 Ack=264 Win=17152 Len=0
www.google.com	192.168.0.11	TLSv1.2	449	Application Data
www.google.com	192.168.0.11	TLSv1.2	703	Application Data
www.google.com	192.168.0.11	TLSv1.2	308	Application Data
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1566 Ack=1562 Win=17408 Len=0
www.google.com	192.168.0.11	TLSv1.2	100	Application Data
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1566 Ack=1608 Win=17408 Len=0
192.168.0.11	www.google.com	TLSv1.2	100	Application Data

You can tell the conversation occurred over a secure protocol in two ways: the protocol is listed as TLS, and the description of the data in the “Info” column only reads “Application Data.” Perhaps a third hint might be the final message sent from my local machine was directed to port 443 of the **www.google.com** server.

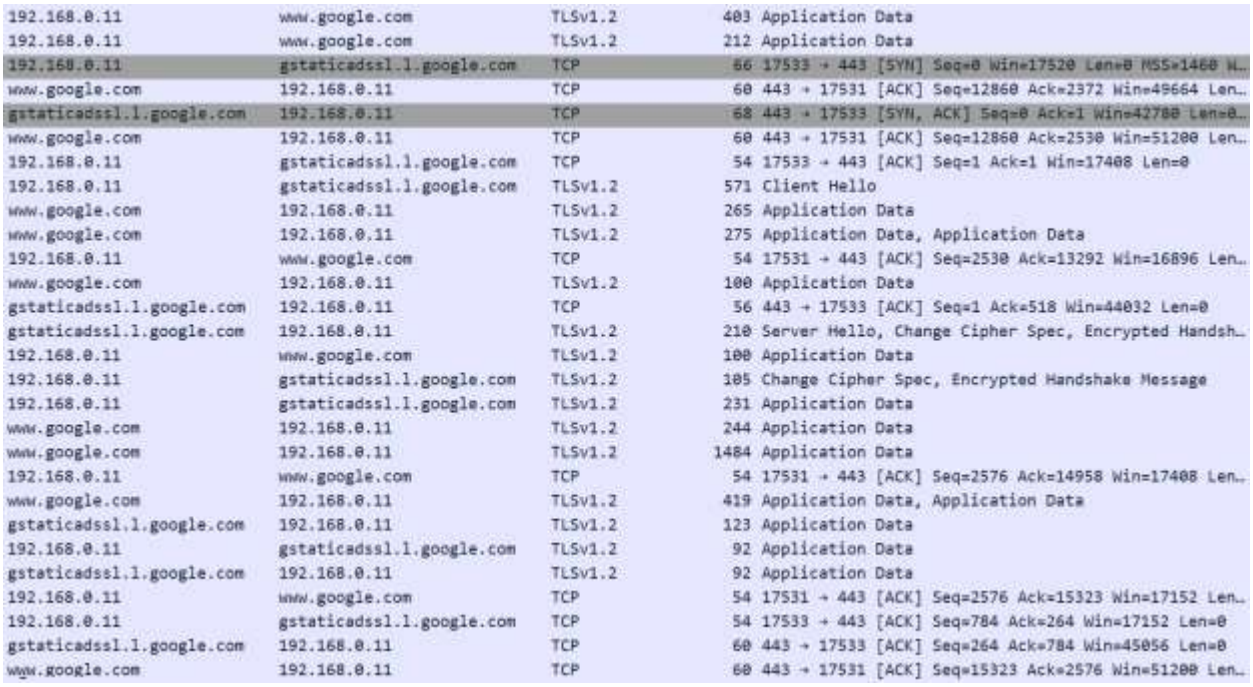
A second conversation occurred among my machine, **www.google.com** and a second Google server at **www.gstatic.com**:

www.gstatic.com	192.168.0.11	TCP	68	443 → 17532 [SYN, ACK] Seq=0 Ack=1 Win=42780 Len=0…
192.168.0.11	www.gstatic.com	TCP	54	17532 → 443 [ACK] Seq=1 Ack=1 Win=17408 Len=0
192.168.0.11	www.gstatic.com	TLSv1.2	246	Client Hello
www.google.com	192.168.0.11	TLSv1.2	355	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	110	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1977 Ack=7685 Win=17408 Len=0
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	1484	Application Data
www.google.com	192.168.0.11	TLSv1.2	893	Application Data, Application Data
192.168.0.11	www.google.com	TCP	54	17531 → 443 [ACK] Seq=1977 Ack=12814 Win=17408 Len…
www.google.com	192.168.0.11	TLSv1.2	100	Application Data
192.168.0.11	www.google.com	TLSv1.2	100	Application Data
www.gstatic.com	192.168.0.11	TCP	56	443 → 17532 [ACK] Seq=1 Ack=193 Win=44032 Len=0



www.gstatic.com	192.168.0.11	TLSv1.2	1484 Server Hello
www.gstatic.com	192.168.0.11	TCP	1484 443 → 17532 [ACK] Seq=1431 Ack=193 Win=44032 Len=1…
www.gstatic.com	192.168.0.11	TLSv1.2	1320 Certificate, Server Key Exchange, Server Hello Done
192.168.0.11	www.gstatic.com	TCP	54 17532 → 443 [ACK] Seq=193 Ack=4127 Win=17408 Len=0
192.168.0.11	www.gstatic.com	TLSv1.2	147 Client Key Exchange, Change Cipher Spec, Encrypted…
192.168.0.11	www.gstatic.com	TLSv1.2	231 Application Data
www.google.com	192.168.0.11	TCP	60 443 → 17531 [ACK] Seq=12860 Ack=2023 Win=48128 Len…
www.gstatic.com	192.168.0.11	TLSv1.2	338 New Session Ticket, Change Cipher Spec, Encrypted …
www.gstatic.com	192.168.0.11	TLSv1.2	123 Application Data
192.168.0.11	www.gstatic.com	TCP	54 17532 → 443 [ACK] Seq=463 Ack=4480 Win=17152 Len=0
192.168.0.11	www.gstatic.com	TLSv1.2	92 Application Data
www.gstatic.com	192.168.0.11	TLSv1.2	92 Application Data
192.168.0.11	www.gstatic.com	TCP	54 17532 → 443 [ACK] Seq=501 Ack=4518 Win=16896 Len=0
www.gstatic.com	192.168.0.11	TCP	60 443 → 17532 [ACK] Seq=4518 Ack=501 Win=45056 Len=0

At this point, conversation with **www.gstatic.com** ended, and the server at **www.google.com** brought a server at **gstaticadssl.l.google.com** into the mix.

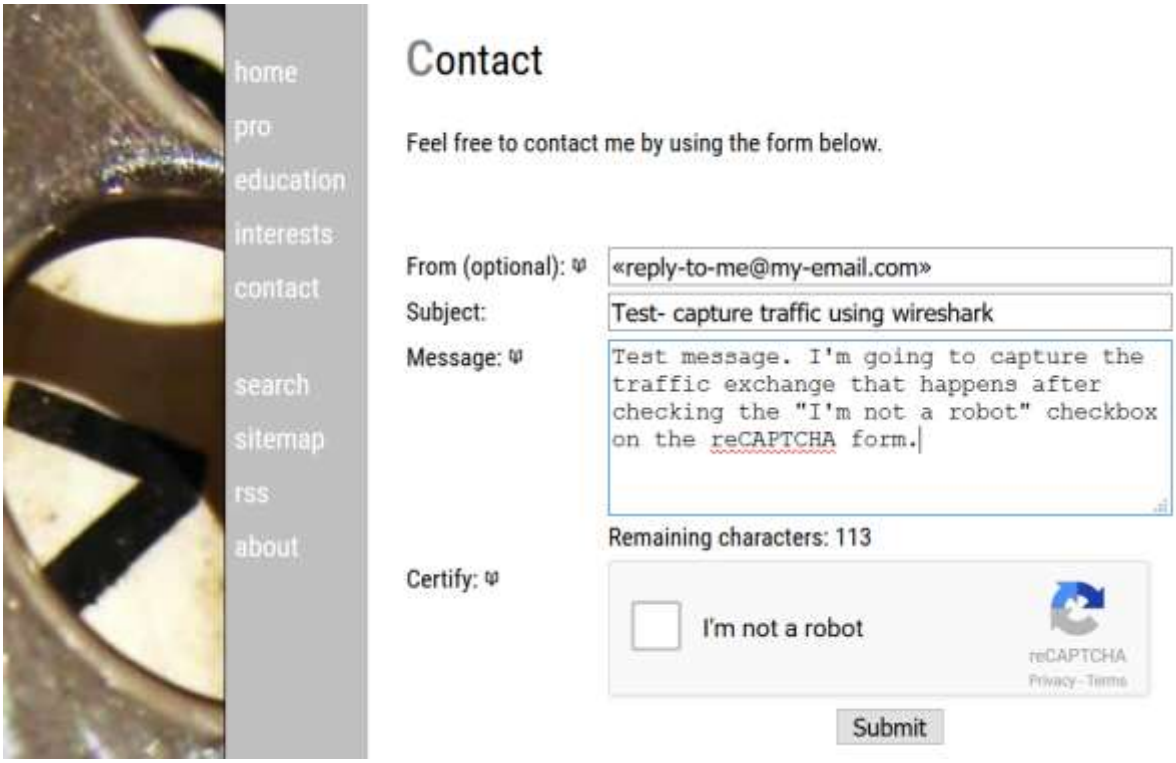


192.168.0.11	www.google.com	TLSv1.2	403 Application Data
192.168.0.11	www.google.com	TLSv1.2	212 Application Data
192.168.0.11	gstaticadssl.l.google.com	TCP	66 17533 → 443 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 W…
www.google.com	192.168.0.11	TCP	60 443 → 17531 [ACK] Seq=12860 Ack=2372 Win=49664 Len…
gstaticadssl.l.google.com	192.168.0.11	TCP	68 443 → 17533 [SYN, ACK] Seq=0 Ack=1 Win=42780 Len=0…
www.google.com	192.168.0.11	TCP	60 443 → 17531 [ACK] Seq=12860 Ack=2530 Win=51200 Len…
192.168.0.11	gstaticadssl.l.google.com	TCP	54 17533 → 443 [ACK] Seq=1 Ack=1 Win=17408 Len=0
192.168.0.11	gstaticadssl.l.google.com	TLSv1.2	571 Client Hello
www.google.com	192.168.0.11	TLSv1.2	265 Application Data
www.google.com	192.168.0.11	TLSv1.2	275 Application Data, Application Data
192.168.0.11	www.google.com	TCP	54 17531 → 443 [ACK] Seq=2530 Ack=13292 Win=16896 Len…
www.google.com	192.168.0.11	TLSv1.2	100 Application Data
gstaticadssl.l.google.com	192.168.0.11	TCP	56 443 → 17533 [ACK] Seq=1 Ack=518 Win=44032 Len=0
gstaticadssl.l.google.com	192.168.0.11	TLSv1.2	210 Server Hello, Change Cipher Spec, Encrypted Handsh…
192.168.0.11	www.google.com	TLSv1.2	100 Application Data
192.168.0.11	gstaticadssl.l.google.com	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
192.168.0.11	gstaticadssl.l.google.com	TLSv1.2	231 Application Data
www.google.com	192.168.0.11	TLSv1.2	244 Application Data
www.google.com	192.168.0.11	TLSv1.2	1484 Application Data
192.168.0.11	www.google.com	TCP	54 17531 → 443 [ACK] Seq=2576 Ack=14958 Win=17408 Len…
www.google.com	192.168.0.11	TLSv1.2	419 Application Data, Application Data
gstaticadssl.l.google.com	192.168.0.11	TLSv1.2	123 Application Data
192.168.0.11	gstaticadssl.l.google.com	TLSv1.2	92 Application Data
gstaticadssl.l.google.com	192.168.0.11	TLSv1.2	92 Application Data
192.168.0.11	www.google.com	TCP	54 17531 → 443 [ACK] Seq=2576 Ack=15323 Win=17152 Len…
192.168.0.11	gstaticadssl.l.google.com	TCP	54 17533 → 443 [ACK] Seq=784 Ack=264 Win=17152 Len=0
gstaticadssl.l.google.com	192.168.0.11	TCP	60 443 → 17533 [ACK] Seq=264 Ack=784 Win=45056 Len=0
www.google.com	192.168.0.11	TCP	60 443 → 17531 [ACK] Seq=15323 Ack=2576 Win=51200 Len…

Conversations among all participants completed in one-third of a second.

Checking the reCAPTCHA Form

The next significant action occurs once the contact form is filled out and the user checks the “I’m not a robot” box on the form, but before submitting the form.



Summary

In this instance, checking the reCAPTCHA form box created about 0.75 seconds of traffic between my local machine and two Google servers – both members of the **1e100.net** domain (I immediately think of "Leeloo").

Details

The first conversation is with a server initially identified by its IP address of 173.194.202.105, and later identified as **pf-in-f105.1e100.net**.

Source	Destination	Protocol	Length	Info
192.168.0.11	173.194.202.105	TLSv1.2	397	Application Data
192.168.0.11	173.194.202.105	TLSv1.2	834	Application Data
192.168.0.11	173.194.202.105	TCP	1434	17967 → 443 [ACK] Seq=1124 Ack=1 Win=258 Len=1380 [TCP segment of a …
192.168.0.11	173.194.202.105	TLSv1.2	431	Application Data
173.194.202.105	192.168.0.11	TCP	56	443 → 17967 [ACK] Seq=1 Ack=344 Win=464 Len=0
173.194.202.105	192.168.0.11	TCP	56	443 → 17967 [ACK] Seq=1 Ack=1124 Win=475 Len=0
173.194.202.105	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=1 Ack=2504 Win=485 Len=0
173.194.202.105	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=1 Ack=2881 Win=496 Len=0
173.194.202.105	192.168.0.11	TLSv1.2	296	Application Data
173.194.202.105	192.168.0.11	TLSv1.2	1484	Application Data
192.168.0.11	173.194.202.105	TCP	54	17967 → 443 [ACK] Seq=2881 Ack=1673 Win=258 Len=0

173.194.202.105	192.168.0.11	TLSv1.2	1484	Application Data
173.194.202.105	192.168.0.11	TLSv1.2	1484	Application Data
173.194.202.105	192.168.0.11	TLSv1.2	1484	Application Data
173.194.202.105	192.168.0.11	TLSv1.2	1484	Application Data
173.194.202.105	192.168.0.11	TLSv1.2	127	Application Data
192.168.0.11	173.194.202.105	TCP	54	17967 → 443 [ACK] Seq=2881 Ack=7466 Win=258 Len=0
173.194.202.105	192.168.0.11	TLSv1.2	100	Application Data
192.168.0.11	173.194.202.105	TCP	54	17967 → 443 [ACK] Seq=2881 Ack=7512 Win=258 Len=0
192.168.0.11	173.194.202.105	TLSv1.2	100	Application Data

The second conversation is with a server initially identified by its IP address of 172.217.9.131, and later identified as **dfw25s26-in-f3.1e100.net**.

Source	Destination	Protocol	Length	Info
192.168.0.11	172.217.9.131	TCP	66	18093 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
173.194.202.105	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=7512 Ack=2927 Win=496 Len=0
172.217.9.131	192.168.0.11	TCP	66	443 → 18093 [SYN, ACK] Seq=0 Ack=1 Win=42780 Len=0 MSS=1380 SACK_PER…
192.168.0.11	dfw25s26-in-f3.1e100.net	TCP	54	18093 → 443 [ACK] Seq=1 Ack=1 Win=66048 Len=0
192.168.0.11	dfw25s26-in-f3.1e100.net	TLSv1.2	248	Client Hello
dfw25s26-in-f3.1e100.net	192.168.0.11	TCP	56	443 → 18093 [ACK] Seq=1 Ack=195 Win=44032 Len=0
dfw25s26-in-f3.1e100.net	192.168.0.11	TLSv1.2	1484	Server Hello
dfw25s26-in-f3.1e100.net	192.168.0.11	TCP	1484	443 → 18093 [ACK] Seq=1431 Ack=195 Win=44032 Len=1430 [TCP segment o…
dfw25s26-in-f3.1e100.net	192.168.0.11	TLSv1.2	1319	Certificate, Server Key Exchange, Server Hello Done
192.168.0.11	dfw25s26-in-f3.1e100.net	TCP	54	18093 → 443 [ACK] Seq=195 Ack=4126 Win=66048 Len=0
192.168.0.11	dfw25s26-in-f3.1e100.net	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
192.168.0.11	dfw25s26-in-f3.1e100.net	TLSv1.2	231	Application Data
dfw25s26-in-f3.1e100.net	192.168.0.11	TLSv1.2	338	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
dfw25s26-in-f3.1e100.net	192.168.0.11	TLSv1.2	123	Application Data
dfw25s26-in-f3.1e100.net	192.168.0.11	TLSv1.2	92	Application Data
192.168.0.11	dfw25s26-in-f3.1e100.net	TCP	54	18093 → 443 [ACK] Seq=465 Ack=4517 Win=65792 Len=0
192.168.0.11	dfw25s26-in-f3.1e100.net	TLSv1.2	92	Application Data

At this point, the first server re-enters conversation:

192.168.0.11	pf-in-f105.1e100.net	TLSv1.2	279	Application Data
192.168.0.11	pf-in-f105.1e100.net	TLSv1.2	838	Application Data
192.168.0.11	pf-in-f105.1e100.net	TCP	1434	17967 → 443 [ACK] Seq=3928 Ack=7512 Win=258 Len=1380 [TCP segment of…
192.168.0.11	pf-in-f105.1e100.net	TLSv1.2	614	Application Data
dfw25s26-in-f3.1e100.net	192.168.0.11	TCP	60	443 → 18093 [ACK] Seq=4517 Ack=503 Win=45056 Len=0
pf-in-f105.1e100.net	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=7512 Ack=3152 Win=507 Len=0
pf-in-f105.1e100.net	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=7512 Ack=3928 Win=518 Len=0
pf-in-f105.1e100.net	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=7512 Ack=5868 Win=533 Len=0
pf-in-f105.1e100.net	192.168.0.11	TLSv1.2	298	Application Data
pf-in-f105.1e100.net	192.168.0.11	TLSv1.2	530	Application Data
pf-in-f105.1e100.net	192.168.0.11	TLSv1.2	215	Application Data
192.168.0.11	pf-in-f105.1e100.net	TCP	54	17967 → 443 [ACK] Seq=5868 Ack=8393 Win=255 Len=0
pf-in-f105.1e100.net	192.168.0.11	TLSv1.2	100	Application Data
192.168.0.11	pf-in-f105.1e100.net	TCP	54	17967 → 443 [ACK] Seq=5868 Ack=8439 Win=254 Len=0
192.168.0.11	pf-in-f105.1e100.net	TLSv1.2	100	Application Data
pf-in-f105.1e100.net	192.168.0.11	TCP	60	443 → 17967 [ACK] Seq=8439 Ack=5914 Win=533 Len=0

The exchanges lasted approximately 0.7s.

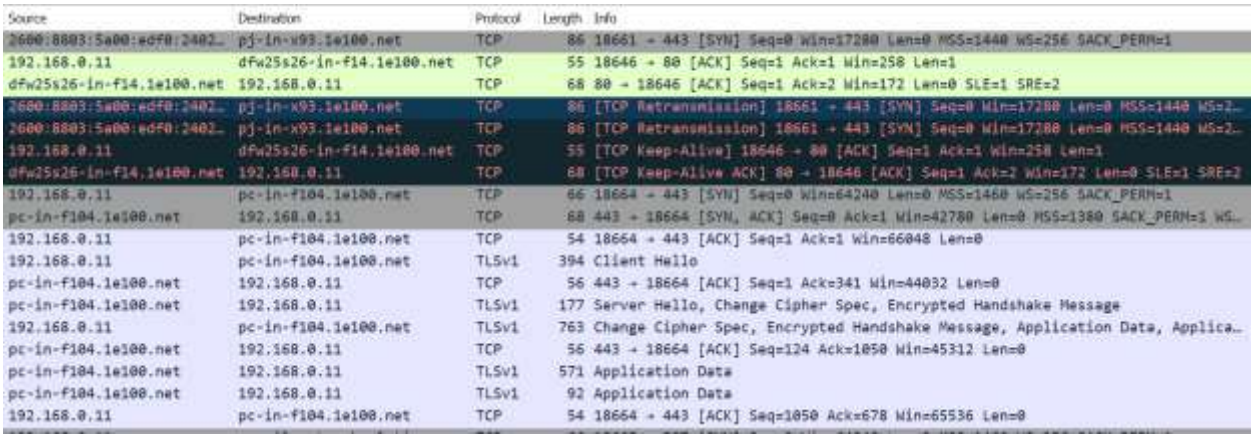
Note that this traffic is the product of just checking that box, with the return of a successful result. Had the control programming determined that more verification was needed, we may have seen much more traffic than this.

Post-Submit

As a function of curiosity, I also looked at the traffic produced after the submit button was clicked. In the code behind, a request of about 600 bytes is sent to Google for the results of the reCAPTCHA validation. A 512-byte response is returned as a stream in JSON format.

Details

"Leeloo" is back for this round, too; notably, **dfw25s26-in-f14.1e100.net** (which is not the same server as before), and **pc-in-f104.1e100.net**. (A third server, **pj-in-x93.1e100.net**, was attempted to be reached over IPv6 but never responded.) Traffic follows:



Source	Destination	Protocol	Length	Info
2600:8803:5a00:edf0:2402…	pj-in-x93.1e100.net	TCP	86	18661 → 443 [SYN] Seq=0 Win=17280 Len=0 MSS=1440 WS=256 SACK_PERM=1
192.168.0.11	dfw25s26-in-f14.1e100.net	TCP	55	18646 → 80 [ACK] Seq=1 Ack=1 Win=258 Len=1
dfw25s26-in-f14.1e100.net	192.168.0.11	TCP	68	80 → 18646 [ACK] Seq=1 Ack=2 Win=172 Len=0 SLE=1 SRE=2
2600:8803:5a00:edf0:2402…	pj-in-x93.1e100.net	TCP	86	[TCP Retransmission] 18661 → 443 [SYN] Seq=0 Win=17280 Len=0 MSS=1440 WS=2…
2600:8803:5a00:edf0:2402…	pj-in-x93.1e100.net	TCP	86	[TCP Retransmission] 18661 → 443 [SYN] Seq=0 Win=17280 Len=0 MSS=1440 WS=2…
192.168.0.11	dfw25s26-in-f14.1e100.net	TCP	55	[TCP Keep-Alive] 18646 → 80 [ACK] Seq=1 Ack=1 Win=258 Len=1
dfw25s26-in-f14.1e100.net	192.168.0.11	TCP	68	[TCP Keep-Alive ACK] 80 → 18646 [ACK] Seq=1 Ack=2 Win=172 Len=0 SLE=1 SRE=2
192.168.0.11	pc-in-f104.1e100.net	TCP	66	18664 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
pc-in-f104.1e100.net	192.168.0.11	TCP	68	443 → 18664 [SYN, ACK] Seq=0 Ack=1 Win=42780 Len=0 MSS=1380 SACK_PERM=1 WS…
192.168.0.11	pc-in-f104.1e100.net	TCP	54	18664 → 443 [ACK] Seq=1 Ack=1 Win=66048 Len=0
192.168.0.11	pc-in-f104.1e100.net	TLSv1	394	Client Hello
pc-in-f104.1e100.net	192.168.0.11	TCP	56	443 → 18664 [ACK] Seq=1 Ack=341 Win=44032 Len=0
pc-in-f104.1e100.net	192.168.0.11	TLSv1	177	Server Hello, Change Cipher Spec, Encrypted Handshake Message
192.168.0.11	pc-in-f104.1e100.net	TLSv1	763	Change Cipher Spec, Encrypted Handshake Message, Application Data, Applica…
pc-in-f104.1e100.net	192.168.0.11	TCP	56	443 → 18664 [ACK] Seq=124 Ack=1050 Win=45312 Len=0
pc-in-f104.1e100.net	192.168.0.11	TLSv1	571	Application Data
pc-in-f104.1e100.net	192.168.0.11	TLSv1	92	Application Data
192.168.0.11	pc-in-f104.1e100.net	TCP	54	18664 → 443 [ACK] Seq=1050 Ack=678 Win=65536 Len=0

Actual conversation between my machine and **pf-in-f104.1e100.net** – during which the request and response are transmitted – spanned approximately 1/3rd of one second.

The traffic that followed this exchange was conversation with the mail server -- the actual email traffic, in SMTP and IMF protocols. That traffic began about 1.5s after the traffic pictured above, and lasted for about 0.7 sec.

Summary

reCAPTCHA will "phone home" to Google on three events:

- When the Web Form is loaded (to retrieve the JavaScript for the API),

- When the “I’m not a robot” control is checked, and
- After the form is submitted, to query the validation results.

When the web form is loaded, a server at **www.google.com** will be queried and the reCAPTCHA API will be retrieved. Other servers involved are at **www.gstatic.com** and **gstaticadssl.l.google.com**.

Upon checking the “I’m not a robot” box on the reCAPTCHA control, you may expect communications with two Google servers located on the **1e100.net** domain lasting about three-quarters of one second (though you can expect this traffic to continue longer than this if additional validation is required).

Finally, after the form is submitted, a request is sent back to Google for the results of the reCAPTCHA validation. The response is returned as a stream. Two servers on the **1e100.net** domain are involved, with traffic lasting approximately 1/3rd of a second. This traffic precedes the actual SMTP traffic for which the reCAPTCHA control acts as a gatekeeper.

Conclusion

I used Wireshark traffic analysis software to explore reCAPTCHA-related network activity in three events of the page lifecycle: when a web form containing the control is loaded, when the reCAPTCHA control is clicked, and when the web form is submitted, and identified several servers involved in the exchange of traffic between Google and the client.

I hope this document offered some idea of what traffic is being generated “under the hood” of version 2 of the Google reCAPTCHA control, and among whom.