Google reCAPTCHA Verification With Sideband Connections

George Watkins, 2012-13-01

Introduction

Virtually every dynamic site on the Internet these days makes use of a CAPTCHA in some fashion. A CAPTCHA is used to verify that a human is driving the interaction with a particular function on a site. A CAPTCHA in its simplest form involves an end-user copying the text from an image to a text field. If the user-entered text matches that of the image, the user is allowed access to the requested resource. Variations to the classic CAPTCHA can involve doing simple math, solving puzzles, or sometimes for accessibility reasons, an audio clip is dictated by the user.

Captchas are usually implemented as part of the application. The classic situation is that particular resource within the application is protected by a CAPTCHA, if an unauthorized user tries to access it, they'll be presented with the CAPTCHA challenge. Once the user enters the correct CAPTCHA it will be noted in their session state and they are free to proceed as normal. Now this mechanism has an obvious issue: the user (or bot) is able to interact directly with the application server even if they are not yet authorized.

When we began working on this tech tip our goal was to decouple the application from the CAPTCHA challenge. We want to be able to prevent the user from ever making contact with our application server until they are authorized. So now the question is: how do we go about this? If there is a BIG-IP already in the infrastructure we could use it to serve our challenge and authorize our users. While iRules is the most advanced traffic processing and manipulation platform around today, it does not shine at generating images. This left us scratching our heads for a few days trying to figure out how we would generate challenge efficiently and dynamically. Enter Google reCAPTCHA.

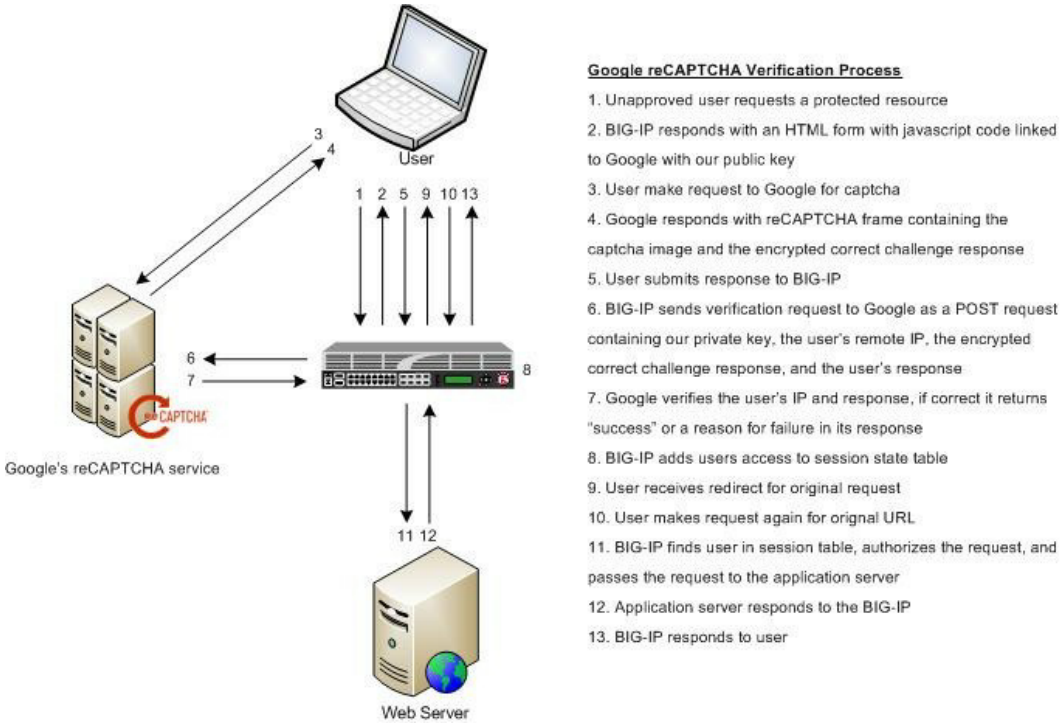
Google reCAPTCHA

Google's reCAPTCHA project is a cleanly packaged CAPTCHA API that operates entirely over HTTP. It was developed to perform two important functions: verifying human interaction and human-operated OCR (optical character recognition). While CAPTCHAs are an annoyance to some users, Google is putting all that seemingly wasted dictation to good use. Current print works undergoing digitization by the reCAPTCHA project include The New York Times and the e-Books offered by Google Books (a portion of which are free). Google estimates that 150,000 work hours each day are spent solving CAPTCHAs. reCAPTCHA is an exemplary model of crowdsourcing in practice.

While the goodwill and crowdsourcing aspects of reCAPTCHA could lead to late night philosophical conversations about the future of the Internet, we are more interested in implementing the API in iRules. Let's begin by discussing how we go about serving a CAPTCHA challenge, ingesting the response, verifying the response, and finally providing passage or denial to our user. This mechanism is facilitated by an key pair obtained from Google.

When an unapproved user accesses a protected resource he will receive the CAPTCHA page containing a form which has a small amount of code provided by Google. Within that code there is javascript and an iframe (for users without javascript). The source links of the javascript and iframe reference the public key provided by Google as a GET variable. The CAPTCHA will then be served by Google with the image and an important hidden form field named "recaptcha_challenge_field." The "recaptcha_challenge_field" hidden field contains the correct response to the CAPTCHA encrypted with your public key. The user then type his response to the CAPTCHA in to the "recaptcha_response_field" text field and submits it for verification. The server receives the response request and assembles a POST request containing the private key (privatekey), the user's IP (remoteip), the encrypted challenge field (challenge), and the user's response (response). These four fields are submitted to Google and the response is verified. Google can return any number of reasons for failure (bad request, bad keys, incorrect response, unmatched remote IP, etc.), but will return "success" in the body if everything proceeds correctly.

Below is a diagram of all the steps involved in presenting and verifying a CAPTCHA challenge with a BIG-IP:



Implementing reCAPTCHA Verification With Sideband Connection

Sideband connections for iRules were introduced in BIG-IP version 11. This functionality opened up a whole new realm of possibilities for iRules including the functionality needed to perform reCAPTCHA verification. We actually attempted this tech tip a year and half ago and hit a road block when we were unable to initiate connection from the BIG-IP. Now performing this verification is a cinch.

The bulk of the work involves assembling a POST request for Google. We tinkered with our request headers until they were reduced to the bare essentials to get a response from Google. These headers include Host (www.google.com), Accept (*/*), Content-length (calculated by taking the string length of the POST payload), and Content-type (application/x-www-form-urlencoded). These headers are concatenated then the POST data payload is appended.

```
1: # extract the encrypted challenge and response from GET request
2: set recaptcha_challenge_field [URI::query [HTTP::uri] "recaptcha_challenge_field"]
3: set recaptcha_response_field [URI::query [HTTP::uri] "recaptcha_response_field"]
4:
5: # assemble body of reCAPTCHA verification POST
6: set recaptcha_post_data "privatekey=$static::recaptcha_private_key&"
7: append recaptcha_post_data "remoteip=[IP::remote_addr]&"
8: append recaptcha_post_data "challenge=$recaptcha_challenge_field&"
9: append recaptcha_post_data "response=$recaptcha_response_field"
10:
11: # calculate Content-length header value
12: set recaptcha_post_content_length [string length $recaptcha_post_data]
13:
14: # assemble reCAPTCHA verification POST request
15: set recaptcha_verify_request "POST /recaptcha/api/verify HTTP/1.1\r\n"
16: append recaptcha_verify_request "Host: www.google.com\r\n"
17: append recaptcha_verify_request "Accept: */*\r\n"
18: append recaptcha_verify_request "Content-length: $recaptcha_post_content_length\r\n"
19: append recaptcha_verify_request "Content-type: application/x-www-form-urlencoded\r\n\r\n"
20: append recaptcha_verify_request "$recaptcha_post_data"
```

```
append recaptcha_verify_request "$recaptcha_post_data
```

Next we'll want to do a lookup for www.google.com and take the first element from the list.

```
# resolve Google's IP address and stuff it into a variable
set google_ip [lindex [RESOLV::lookup @$static::dns_server -a "www.google.com"] 0]
```

Then we open a connection to Google, send the POST request, then wait for a response.

```
# establish connection to Google
set conn [connect -timeout 1000 -idle 30 $google_ip:80]

# send reCATPCHA verification request to Google
send -timeout 1000 -status send_status $conn $recaptcha_verify_request
```

Finally, if we receive a response before the timeout, we parse it and match on "success." If the request times out the user will see another CAPTCHA challenge. If we do get a successful response we'll add the user status to a session table for a period of time and redirect them to their original request.

```
# receive reCAPTCHA verification response from Google
set recaptcha_verify_response [recv -timeout 1000 -status recv_info $conn]

# close connection
close $conn

# process reCAPTCHA verification response and remove user session from trigger table if successful
if { $recaptcha_verify_response contains "success" } {
set redirect_url [table lookup -subtable $static::recaptcha_redirect_table -notouch $session_identifier]

table add -subtable $static::recaptcha_approval_table $session_identifier 1 \
$static::recaptcha_approval_timeout $static::recaptcha_approval_lifetime

HTTP::redirect $redirect_url
} else {
HTTP::respond 200 content $static::recaptcha_challenge_form
}
```

If you would like additional information on sideband connections, Colin did an excellent write-up on them. The iRules wiki is quickly becoming populated with sideband connection examples as well.

Google reCAPTCHA Challenge iRule

The reCAPTCHA is largely a proof of concept. It is purposely generic so that it can be easily modified for anyone's use case. It currently does not provisions for triggering on specific URLs or connection conditions. Adding these or other condition-specific triggers should be as easy as capping the code with if statements to match URLs or other conditions.

A few static variables are located at the top of the iRule that will need to be configured prior to deployment:

dns_server - DNS server address for which the BIG-IP has a direct route
recaptcha_public_key - reCAPTCHA public key obtained from Google; Google reCAPTCHA Admin Panel
recaptcha_private_key - reCAPTCHA private key obtained from Google; Google reCAPTCHA Admin Panel
recaptcha_approval_table - tracks CAPTCHA challenge approvals by user IP and source port; default = "recaptcha_approvals"
recaptcha_redirect_table - maintains redirect state so user sees their previously requested resource after the CAPTCHA challenge; default = "recaptcha_redirects"
recaptcha_approval_timeout - timeout of CAPTCHA approval; default = 3600s
recaptcha_approval_lifetime - lifetime of CAPTCHA approval; default = 3600s
debug - debug level, 0 = silent, 1 = log client interaction, 2 = log all interaction with client and Google

The full source for the Google reCAPTCHA Challenge iRule can be download from the DevCentral CodeShare.

Conclusion

CAPTCHAs are a great way to protect a portion of your site from bots. Everything from new user signups to ticket purchases make use of CAPTCHAs these days. As the overhead for implementation becomes lower, their use become more prevalent. The main barrier to entry is modifying an application to perform all the logic involved in triggering, presenting, and verifying the CAPTCHA. With a BIG-IP and iRules, the application no longer needs to be modified. The CAPTCHA challenge form can be presented from the BIG-IP or as a static page on a server and all of the verification and processing takes place outside of the application server. If you're looking for an easy way to implement CAPTCHAs for your site, look no further than BIG-IP version 11 and the Google reCAPTCHA Challenge iRule.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com