Google reCAPTCHA Challenge

Use Case:

Over the years client authentication has become multi-fold due to the risks involved from various factors. While the application owners worry about valid client authentication, the availability of the service itself becomes a question due to possible DoS attacks against the service. Many times there are automated bots trying to guess the passwords too.

CAPTCHA provides the ability to take interactive user feedback in real time. It ensures that no automated bot can try and guess the passwords over number of attempts and it also ensures that user accounts are not locked out by trying wrong passwords repeatedly. Google provides this service which can be integrated in our ADC authentication flow and validated with Google in real time.

F5 iRule:

```
when RULE_INIT {
  # set DNS server address to resolve www.google.com
  set static::dns_server 10.0.0.254

  # set public and private reCAPTCHA keys (obtain from www.recaptcha.com)
  set static::recaptcha_public_key "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  set static::recaptcha_private_key "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

  # set name of table to track reCAPTCHA approvals
  set static::recaptcha_approval_table "recaptcha_approvals"

  # set name of table to track last requested URL
  set static::recaptcha_redirect_table "recaptcha_redirects"

  # timeout and/or lifetime for reCAPTCHA approval
  set static::recaptcha_approval_timeout 3600
  set static::recaptcha_approval_lifetime 3600
```

```
  # log level, 0 = silent, 1 = log client interaction, 2 = log all inte
raction with client and Google
  set static::logging 1


  # begin - HTML for reCAPTCHA form page
  set static::recaptcha_challenge_form {<html>
  <head>
    <title>Hold up there!</title>
  </head>
  <body>
    <div align="center">
      <h1>Hold up there!</h1>
    <h3>We want to make sure that you're a human before you can use thi
s virtual server!</h3>
    <form action="/verify_recaptcha" method="GET">
        <script type="text/javascript"
          src="http://www.google.com/recaptcha/api/challenge?k=}
    append static::recaptcha_challenge_form $static::recaptcha_public_k
ey
  append static::recaptcha_challenge_form {">
        </script>
        <noscript>
          <iframe src="http://www.google.com/recaptcha/api/noscript?k=}
  append static::recaptcha_challenge_form $static::recaptcha_public_key
  append static::recaptcha_challenge_form {"
            height="300" width="500" frameborder="0"></iframe>
          <textarea name="recaptcha_challenge_field" rows="3" cols="40"
>
          </textarea>
          <input type="hidden" name="recaptcha_response_field" value="m
anual_challenge">
        </noscript>
        <input type="submit" value="Submit">
      </form>
  </div>
  </body>
```

```
</html>}

  # end - HTML for reCAPTCHA form page

}


when CLIENT_ACCEPTED {

  set session_identifier "[IP::client_addr]:[TCP::client_port]/[IP::local_addr]:[TCP::local_port]"

}


when HTTP_REQUEST {

  if { [HTTP::path] equals "/verify_recaptcha" } {

    set recaptcha_challenge_field [URI::query [HTTP::uri] "recaptcha_challenge_field"]

    set recaptcha_response_field [URI::query [HTTP::uri] "recaptcha_response_field"]


    if { $static::logging >= 1 } {

      log local0. "Session \"$session_identifier\": user responded with \"$recaptcha_response_field\""

    }


    # assemble body of reCAPTCHA verification POST

    set recaptcha_post_data "privatekey=$static::recaptcha_private_key&"

    append recaptcha_post_data "remoteip=[IP::remote_addr]&"

    append recaptcha_post_data "challenge=$recaptcha_challenge_field&"

    append recaptcha_post_data "response=$recaptcha_response_field"


    # calculate Content-length header value

    set recaptcha_post_content_length [string length $recaptcha_post_data]


    # assemble reCAPTCHA verification POST request

    set recaptcha_verify_request "POST /recaptcha/api/verify HTTP/1.1\r\n"

    append recaptcha_verify_request "Host: www.google.com\r\n"

    append recaptcha_verify_request "Accept: */*\r\n"
```

```
    append recaptcha_verify_request "Content-length: $recaptcha_post_co
ntent_length\r\n"

    append recaptcha_verify_request "Content-type: application/x-www-fo
rm-urlencoded\r\n\r\n"

    append recaptcha_verify_request "$recaptcha_post_data"


    # 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]


    # 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_requ
est


    if { $static::logging >= 2 } {

      log local0. "Session \"$session_identifier\": sending reCAPTCHA v
erification to Google: \"$recaptcha_verify_request\""

    }


    # receive reCAPTCHA verification response from Google

    set recaptcha_verify_response [recv -timeout 1000 -status recv_info
$conn]


    if { $static::logging >= 2 } {

      log local0. "Session \"$session_identifier\": received verificati
on response from Google: \"$recaptcha_verify_reponse\""

    }


    close $conn


    # process reCAPTCHA verification response and remove user session f
rom trigger table if successful

    if { $recaptcha_verify_response contains "success" } {
```

```
      set redirect_url [table lookup -subtable $static::recaptcha_redir
ect_table -notouch $session_identifier]


      table add -subtable $static::recaptcha_approval_table $session_id
entifier 1 \
        $static::recaptcha_approval_timeout $static::recaptcha_approval
_lifetime


      if { $static::logging >= 1 } {
        log local0. "Session \"$session_identifier\": passed captcha an
d was added to approval table"
        log local0. "Session \"$session_identifier\": redirecting to \"
$redirect_url\""
      }


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


      if { $static::logging >= 1 } {
        log local0. "Session \"$session_identifier\": failed captcha"
      }
    }
  } elseif { [table lookup -subtable $static::recaptcha_approval_table
$session_identifier] eq ""} {
    table add -subtable $static::recaptcha_redirect_table $session_iden
tifier "http://[HTTP::host][HTTP::uri]" 3600 3600
    HTTP::respond 200 content $static::recaptcha_challenge_form
  }
}
```

Netscaler Solution:

#Compute SessionID

add policy expression session_id
"client.ip.src+\":\"+client.tcp.srcport+\":\"+client.ip.dst+\":\"+client.tcp.dstport"

```
#Callout to google.com to validate recaptcha response
add policy httpCallout hc1 -IPAddress 74.125.203.105 -port 443 -returnType TEXT -hostExpr
"\"www.google.com\"" -urlStemExpr
"\"/recaptcha/api/siteverify\?secret=6LcEqCATAAAAADljVadulW6RIXNawLg3J86WN2tW&re
sponse=\"+http.req.body(10000).after_str(\"g-recaptcha-response=\")" -scheme https -
resultExpr "http.res.body(1000000).XPATH_json(xp%//success%)"

#Map to store the SessionID and the response
add ns variable google_map -type "map(text(1000),text(100), 100)" -expires 100
add ns assignment assign_google_map -variable "$google_map[session_id]" -set
"sys.http_callout(hc1)"

#Replace to actual url after validation of recaptcha
add rewrite action replace_url replace http.req.url
"http.req.header(\"Referer\").typecast_http_url_t"
add rewrite policy pol_replace_url "$google_map[session_id].contains(\"true\")"
replace_url

#Insert a cookie in the response so that next time same user is not redirected to google
recaptcha
add rewrite action insert_session_id insert_http_header Set-Cookie
"\"Session_ID=\"+session_id"
add rewrite policy pol_session_id
"$google_map[session_id].set_text_mode(ignorecase).contains(\"true\")" insert_session_id

#Reset connection if something goes wrong
add rewrite policy pol_false
"$google_map[session_id].set_text_mode(ignorecase).eq(\"false\")" RESET

#Initial redirection to google recaptcha page
#The verify page should be imported to NS to make the below config work.
#It can also be copied to /var/tmp and import from local
#Please find the html page after the configs
add responder action respond_verify respondwithhtmlpage verify
add responder policy pol_respond_verify
"http.req.cookie.value(\"Session_ID\").length.eq(0) &&
http.req.url.query.value(\"verify_recaptcha\").eq(\"true\").not" respond_verify

#Assign SessionID and response in Variable
add responder policy rs_pol_assign
"http.req.url.query.value(\"verify_recaptcha\").eq(\"true\")" assign_google_map

#Policy Bindings
bind lb vserver vip1 -policyName pol_replace_url -priority 3 -gotoPriorityExpression next -
type REQUEST
```

```
bind lb vserver vip1 -policyName pol_session_id -priority 1 -gotoPriorityExpression next -type RESPONSE
bind lb vserver vip1 -policyName pol_false -priority 2 -gotoPriorityExpression next -type RESPONSE
bind lb vserver vip1 -policyName rs_pol_assign -priority 99 -gotoPriorityExpression END -type REQUEST
bind lb vserver vip1 -policyName pol_respond_verify -priority 100 -gotoPriorityExpression END -type REQUEST
```

Responder HTML Page:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Google ReCaptcha with Netscaler</title>
  <script src='https://www.google.com/recaptcha/api.js'></script>
 </head>

 <body>

  <form action="http://10.102.62.182/${http.req.url}?verify_recaptcha=true" method="post">

   <H1> We want to make sure that you're a human before you can use this virtual server! </H1>

    <div class="g-recaptcha" data-sitekey="6LcEqCATAAAAANZC7PdKHc9NA6q20MFl9DTdidpC"></div>

    <input type="submit" value="Submit" />

  </form>
</body>
</html>
```

How it Works:

- Each time when a new user tries to login, he is redirected to a Login Page where he takes up the Google Re-Captcha Challenge.
- After completion of the challenge, we send a callout to google to validate the response.
- Based on the response from the Google, we decide whether to allow or block the user
- As an Add-on apart from the iRule, we have added cookie validation. So the returning user within a specified time interval (configurable in map variable) would not be redirected to the recaptcha page and would have access to the actual requested page. This is an option and can be removed if customer does not want this.

- To get this working, we need to generate private and public keys from google.com
- Please refer the url - https://www.google.com/recaptcha/ to generate the keys