FSE iRules Challenge Roundup: They did what now?

Colin Walker, 2013-16-12

There's this thing we do every time we hire a batch of new Field Sales Engineers. We call it Boot Camp. We bring them in for two solid weeks of training and kool-aid drinking before returning them to the wild in their respective corners of the world and cutting them loose. By the time they're done they're bigger, better, faster, stronger…or you know, at least more knowledgeable. As part of that process I get to jump in and try to break their brains with iRules goodness. Really I'm not trying to break them, I suppose, as much as help them figure out what they'll need to know, and how to find it.

The format is simple: Dream up a challenging problem that needs to be solved via an iRule (and potentially other pieces of F5 technology), write it up and present it to the new FSEs as if I were a user. They then get some time to ask questions, and begin working to solve the problem. After the allotted time (usually a week of piecemeal time between training sessions and whatnot) I come back and judge the winners based on accuracy, efficiency and portability/understandability. It's fun for me and I'm hoping helpful for them.

Every time I go through this process I'm impressed with the entries I get back and this time was absolutely not an exception. This was the largest FSE group we've had through to date and I have to say the quality of entries was fantastic. This was a really tough challenge, so much so that after I had presented it and gone back to my desk to figure out how *I* would solve it I stopped, cursed a little under my breath, and thought I may have just set the bar way too high for any of the new fish to jump over. Remember that these people are new employees to the company, usually from days to a couple months at most, and most have zero exposure to programming and iRules beyond whatever they managed to pick up along the way. No formal training, etc. I thought I may have stumped them all given some of the odd complexities of the challenge I laid out. Boy was I wrong. So it is with no small amount of pride that I bring you this installment of the FSE iRules Challenge Roundup.

First, as always, the challenge I issued:

> Scenario:
>
> We're moving to a new web application back end that has some modified requirements from our previous solution. We need to ensure that the change over goes smoothly. To do so we need to patch a couple of business logic issues and add some monitoring to be sure we're getting the modifications we need. Here's what we'd like…
>
> Desired Solution:
>
> 1.Enable cookie based persistence. Check for a cookie named "app-persist" and persist based on the value of that cookie if it exists. If not, add a new cookie with that name with a hashed version of the client IP as the value.
>
> 2.Ensure traffic is routed to the proper pool. The old application ran on a single port, whereas the new one runs on multiple (8080-8090), and treats each differently. Route traffic on port 8080 to pool "http_8080", port 8081 to pool "http_8081", and so on.
>
> 3.Rewrite all occurrences in the server's response from "oldapp1.domain.com" to "domain.com:8080", "oldapp2.domain.com" to "domain.com:8081", etc. up to "oldapp10.domain.com" being rewritten to "domain.com:8090".
>
> 4.Log every server-side rewrite to an off-box logging server. Capture the client's IP, the server's IP, the time, the URL requested and what the rewrite was, I.E. "oldapp1.domain.com -> domain.com:8080".

And then the winners:

Third Place – Ahmad Moubarak

Ahmad had an excellent approach and logical flow with his code. It was apparent that he groked the goals of the challenge right off and was very much on the right track to solving them. He ended up using loops, which I almost never recommend, but he did so in an interesting way. It's obvious he's got some scripting background (unconfirmed assumption) by the way he tackle this. That will serve him well once he learns how coding on high volume systems like a BIG-IP are a bit different than most other environments. He also, along with everyone else that submitted an entry, grabbed hold of the stream profile concept right off the bat. Overall really nice work.

```
when CLIENT_ACCEPTED {
  # Get the Client IP and destination port
    set clientip [IP::client_addr]
    set requestport [TCP::local_port]
  # Hash the STRING format of the Client IP using MD5
    set haship [md5 "$clientip"]
}

when HTTP_REQUEST {
  # List of Application Ports.
  set ports_list [list 8080 8081 8082 8083 8084 8085 8086 8087 8088 8089 8090]
  set server_list [list 1 2 3 4 5 6 7 8 9 10 11]
  set url [HTTP::host]
  STREAM::disable
  # This loop to detect which port is being used and send the request to the corresponding pool only if thecookie has been inserted
    foreach x $ports_list {
        if  {[HTTP::cookie exists "app-persist"] && ($requestport == $x)} {
            pool "http_$x"
            persist uie "$haship"
            set cookie 0
            break
        }
        else {
  # If cookie is not found, then send the first request to the corresponding pool
            foreach x $ports_list {
                if  {($requestport == $x)} {
                pool "http_$x"
                set cookie 1
                break
            }
        }
    }
  }
}

when HTTP_RESPONSE {
  # Insert the hashed value of the client IP as a cookie into the client's browser
  if {$cookie} {HTTP::cookie insert name "app-persist" value "$haship"}
   set serverip [IP::server_addr]
   set initial_value 8079
  # The loop will not break on the first match since I am assuming that the page might return multiple links to domain.com:Port_Number
      foreach n $server_list {
     STREAM::expression {@oldapp$n.domain.com@ @domain.com:[expr $initial_value + $n]@}
    }
    STREAM::enable
  # Logging information to Syslog Server 10.128.10.1
   log -noname 10.128.10.1 "Client IP = $clientip,
                            Server IP = $serverip,
                            The Time is [clock format [clock seconds] -format {%H:%M:%S}],
                            The Requested URL is $url"
}


when STREAM_MATCHED {
```

```
# Logging re-write actions as they happen to Syslog Server 10.128.10.1
    log -noname 10.128.10.1 "Rewrite Action: oldapp$n.domain.com -----> domain.com:[expr $initial_value + $n]"
}
```

Second Place – Pavin Seejuntra

Pavin … well … let's just say Pavin must be some kind of serious workhorse. While more code isn't always better, and frankly I often preach just the opposite, I.E. less is more, I was pretty impressed at the dedication to knocking out this challenge. He took a solid approach and saw it through, even when it required a fair amount of work, and that's commendable. He has a great grasp on the HSL functionality in iRules and some higher level programming concepts that I didn't expect to see in this particular challenge. That, plus the fact that the solution was awfully close to functional, put Pavin in second place. Nice work. Believe it or not I've shortened his code here, so this isn't quite representative of the entire solution he submitted, and isn't quite as functional, but you'll get the general idea at least.

```
when RULE_INIT {
  # Set to 1 to enable debugging to /var/log/ltm
  # Set to 0 to disable debugging
  set static::debug 1
}
when CLIENT_ACCEPTED {
   set hsl [HSL::open -proto UDP -pool syslog_server_pool]
}

when HTTP_REQUEST {
  HTTP::header remove "Accept-Encoding"
  set csid "[IP::client_addr]:[TCP::client_port]"
  set dst_port [TCP::local_port]
  set url [HTTP::header Host][HTTP::uri]
  set vip [IP::local_addr]:[TCP::local_port]
  set http_request_time [clock clicks -milliseconds]
  STREAM::disable

    if {([TCP::local_port] >= 8080) && ([TCP::local_port] <= 8090) } {
        pool http_$dst_port
        if { [HTTP::cookie exists "app-persist"] } {
        if {$static::debug} { log local0. "$csid Cookie: [HTTP::cookie app-persist]" }
            persist uie [HTTP::cookie value "app-persist"]
        }
    } else { pool http_80}
}


when LB_SELECTED {
  if {$static::debug} { log local0.  "$csid Selected server [LB::server]" }
}
when HTTP_RESPONSE {
  set client [IP::client_addr]:[TCP::client_port]
    set node [IP::server_addr]:[TCP::server_port]
    set nodeResp [HTTP::status]

    if {$static::debug} { log local0.  "$csid Cookie Read: [HTTP::cookie names]" }
    if { [HTTP::cookie exists "app-persist"] } {
        if {$static::debug} { log local0.  "$csid Cookie Exist: [HTTP::cookie app-persist]" }
        persist add uie [HTTP::cookie value "app-persist"]
    } else {
        HTTP::cookie insert name "app-persist" value [md5 [IP::client_addr]]
        persist add uie [HTTP::cookie value "app-persist"]
        if {$static::debug} { log local0.  "$csid Cookie Insert: [HTTP::cookie app-persist]" }
    }

    if {[HTTP::header value Content-Type] contains "text"}{
        # Match an http://*example.com string and replace it with nothing yet
        STREAM::expression {&oldapp.*?domain\.com&&}
        # Enable the stream filter for this response only
        STREAM::enable
    }
}

when STREAM_MATCHED {

    if {$static::debug} { log local0.  "$csid Stream:Match: [STREAM::match]" }

    switch [STREAM::match] {

        oldapp1.domain.com {

            STREAM::replace "[string map {oldapp1.domain.com domain.com:8080} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url -> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8080"

        }
        oldapp2.domain.com {

            STREAM::replace "[string map {oldapp2.domain.com domain.com:8081} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url-> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8081"
        }
        oldapp3.domain.com {

            STREAM::replace "[string map {oldapp3.domain.com domain.com:8082} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url-> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8082"

        }
        oldapp4.domain.com {

            STREAM::replace "[string map {oldapp4.domain.com domain.com:8083} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url -> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8083"

        }
        oldapp5.domain.com {

            STREAM::replace "[string map {oldapp5.domain.com domain.com:8084} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url-> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8084"

        }
        oldapp6.domain.com {

            STREAM::replace "[string map {oldapp6.domain.com domain.com:8085} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url-> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8085"

        }
        oldapp7.domain.com {

            STREAM::replace "[string map {oldapp7.domain.com domain.com:8086} [STREAM::match]]"

            HSL::send $hsl "[clock format [clock seconds] -gmt 1 ] Client: $client -> VIP:$vip URL:$url -> Node: $node with response $nodeResp : Replaced: [STREAM::match]->domain.com:8086"

        }
```

Winner! – Alex Tsai

Last but the polar opposite of least, we have Alex with the winning submission. Really, really great work went into getting this nearly functional, largely polished piece of code put together for the challenge this time around. It's compact, it's nearly fully functional, it's elegant, it's easy to follow…frankly it's all the things I would normally look for in an iRule that I was going to deliver to a customer. With this kind of start I can only assume that Alex is going to go on to stomp out some darn impressive iRules when he gets fully up to speed and a little more experience with the language. This is a killer result and I'm duly impressed. Way to set the bar man, and here's hoping to see more work out of you in the future.

```
when CLIENT_ACCEPTED {
        #set hsl [HSL::open -publisher /Common/my_log_publisher]
        set hsl [HSL::open -proto UDP -pool syslog_server_pool] }
}

when HTTP_REQUEST {
if { [HTTP::cookie exists "app-persist"] } {
          persist uie [HTTP::cookie "app-persist"]
          set flag 0
        } else {
          # Set a flag to indicate there is a need to set cookie during HTTP_RESPONSE
          set flag 1
        }
    set url "http://[HTTP::host][HTTP::uri]"

   # Disable the stream filter for all requests
        STREAM::disable
        HTTP::header remove "Accept-Encoding"
}

when HTTP_RESPONSE {
   # Check if response type is text
   if {[HTTP::header value Content-Type] contains "text"}{
          STREAM::expression {@oldapp(\d*)\.domain\.com@@}
          # Enable the stream filter for this response
          STREAM::enable
          # If the flag is set, set cookie to Browser
          if {$flag == 1 } {
                        HTTP::cookie insert name "app-persist" value [md5 [IP::client_addr]]
                        persist add uie [HTTP::cookie "app-persist"]
          }
        }
}

when STREAM_MATCHED {
   # Check if the matched string meets some condition that can't easily be checked for using a single regex in STREAM::expression
        regexp {oldapp(\d*)\.domain\.com$} [STREAM::match] -> m1
        set port [expr 8079+$m1]
        set newstring "domain.com:$port"
        STREAM::replace $newstring
        HSL::send $hsl "Client IP: [IP::client_addr] Server IP: [IP::server_addr] HTTP URL: $url, Replace \"[STREAM::match]\" with \"$newstring\""
        log local0.    "Client IP: [IP::client_addr] Server IP: [IP::server_addr] HTTP URL: $url, Replace \"[STREAM::match]\" with \"$newstring\""
}
```

That does it for this roundup. Great work to all the FSEs that submitted an entry. I love seeing the ingenuity and talent we're adding to the team, and I can't wait to see what everyone's up to next. Keep on iRuling and never hesitate to lean on the resources with which this challenge hopefully helped you get acquainted.

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