20 Lines or Less #63: Port Routing, Binary Scanned Payloads, and Cookie Functionality testing

Colin Walker, 2012-18-09

What could you do with your code in 20 Lines or Less? That's the question I like to ask for the DevCentral community, and every time I go looking to find cool new examples that show just how flexible and powerful iRules can be without getting in over your head.

This week the 20 Lines or Less is brought to you from the road. From Raleigh, North Carolina to be exact. I'm here today for another awesome F5 User Group, and am excited to add a dose of DevCentral and iRules goodness to the agenda. As such it's even more appropriate than normal that I find myself crawling through the forums to see what the good people of DevCentral have been up to recently. As always, I'm far from let down. The community provides example after awesome example of just how iRules can be so powerful and useful with so little effort. This week I've got offerings of port based routing, binary scanning goodness, and an interesting take on cookie inspection. So let's get started!

Route Traffic to specific port via URL

http://bit.ly/PnBiUu

Chase asked about how to do some routing based on URL (host name, really). He wanted to look at the cname of a particular domain and, based on that, route to the same pool of servers on one of two different ports. While there are probably a few other, tricky ways to do this, Kevin came through with the simple, logical approach that I fully support. Two pools, same members, different ports, super simple iRule to select which one you want. Bing-bang-boom, you've got a port switcher based on host information. A simple solution that'll get the job done with barely any code to speak of, and we here at the 20LoL writing desk approve of such things, not shockingly.

```
1: when HTTP_REQUEST {
2:     switch [string tolower [HTTP::host]] {
3:         "appa.domain.com" { pool appa_pool }
4:         "appb.domain.com" { pool appb_pool }
5:     }
6: }
```

TCP Collection and binary scanning

http://bit.ly/QyOqFO

There are approximately 109801284814901284380129850128 32 different reasons that you might want to do some collection within an iRule. This is an example of one of those (no, I don't know which number it is), in which rahtoll was looking to seek out a specific section of a payload and see if it matched a given value. There are a few ways to go about this, and after a little coaching from some of DC's finest in the form of Michael Yates and the indisputably awesome Hoolio, it looks like a solution is now in place. If you've got inspection needs for specific chunks of a payload, this might be a good one to keep in your hip pocket.

```
1: when CLIENT_ACCEPTED {
2:   TCP::collect 65
3: }
4:
5: when CLIENT_DATA {
6:
7:   # use pool2 by default
8:   pool test_pool2
9:
10:   set mero [TCP::payload 15]
11:   # Start reading at the 5th byte and save 6 characters to $match
12:   if {[binary scan $mero x5a6 match] == 1}{
13:     log local0. "Matched $match"
14:     if { $match eq "000001" } {
15:       pool test_pool1
16:     }
17:   }
18:   TCP::release
19: }
```

Cookie Checking

http://bit.ly/SYaNtH

We've covered cookies a dozen or more times over the years in the 20LoL (The fact that I can say that is a bit scary, and hawesome) but this is something I haven't seen before. This is a way to check to see if cookies are enabled in a given browser, and a rather clever one at that. I'm sure there are some use cases for this in production, but there are definitely some in different test environments, and it may be the start to some cool deployment logic. A neat idea at the very least, and something that caught my eye in my perusal of all things iRules.

```
1: when HTTP_REQUEST {
2:     if { ( [HTTP::uri] starts_with "/cookietest" ) } {
3:         ## test for cookie
4:         if { not ( [HTTP::cookie exists "COOKIETEST"] ) } {
5:             ## cookies are disabled
6:             HTTP::respond 200 content "Sorry. This site requires cookies"
7:         } else {
8:             ## cookies enabled - redirect to first URI
9:             HTTP::respond 302 Location [findstr [HTTP::uri] "/cookietest" 11]
10:         }
11:     } elseif { [HTTP::cookie count] == "" } {
12:         ## no cookies (could be first request) - redirect to cookie test URI with test cookie
13:         HTTP::respond 302 Location "/cookietest[HTTP::uri]" "Set-Cookie" "COOKIETEST=1"
14:     }
15: }
```

There you have it, another addition to the heap of killer iRules contributions that can get you a handy fix, solution or idea in less than 21 lines of code. See you in 2 weeks for more 20LoL goodness.

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