20 Lines or Less # 42 – Secret list … OF DOOM

Colin Walker, 2010-09-12

What could you do with your code in 20 Lines or Less? That's the question I ask (almost) every week for the devcentral community, and every week 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 I bring to you two cool iRules from the forums and one from a secret squirrel internal mailing list OF DOOM!!!. Okay, maybe not of doom, but it's a list with some wicked smart people on it and I'm lucky to get to harvest good stuff out of there sometimes. We'll cover handling HTTP and HTTPS on a single VIP, stream matching TCP data, and HTTP rate limiting with the table command in yet another variation.

TCP Payload Matching and Logging

http://bit.ly/efMH9z

User mattrm was kind enough to post back to this thread started a few months back to show us exactly how he got this stream match solution working in his environment. A very cool example of how to search and replace a string in-line. In this case he's dealing with straight TCP traffic, but just about any non encrypted traffic (including traffic decoded on the LTM) would work. Using the stream profile: @Username.*Userpassword=\n@@ :

```
when STREAM_MATCHED {
  # log each match found by the stream filter
  scan [STREAM::match] "Username=%s" user
  log local0. "Username [b64decode $user] had password replaced"
  STREAM::replace "Username=$user\nUserpassword=0000=\n"
}


when LB_SELECTED  {
set serverIP [LB::server addr]
log local0. "LB Server IP $serverIP"
}
```

Selective HTTP redirection based on SSL

http://bit.ly/g6bCbG

In this interesting iRule user Dave Hatfield is looking to redirect traffic that is not using SSL to an HTTPS link, while dealing with non SSL traffic differently, but on the same virtual and in the same iRule. That poses an interesting problem of identifying SSL vs non SSL traffic in a single virtual / iRule. There are a couple of ways to go about this, and I think he chose a solid one. Simple and easy to follow, yet effective. Cool stuff.

```
when CLIENT_ACCEPTED {
        # Set a variable to track whether this is an HTTPS request
        set https 0
}
when CLIENTSSL_HANDSHAKE {
        # There was a client side SSL handshake, so update the
variable
        set https 1
}
when HTTP_REQUEST {
        # Check if referrer is eschool
        if {!([HTTP::header "Referer"] starts_with
"https://eschool.mysite.org/webapps/")}{
              # If there is no URI redirect to portal
             if {[HTTP::path] eq "/" }{
                   HTTP::redirect "https://portal.mysite.org/eschool"
             }
}
        if {not ($https)}{
        # If it's not an HTTPS connection, send a redirect
              HTTP::redirect https://[HTTP::host][HTTP::uri]
        }
}
```

HTTP Request limiting via tables

One of the engineers here at F5, Christian Koenning spawned a cool example iRule showing off how to achieve a stable 3000 requests per second threshold by making use of the table command and some logic inside an iRule. The illustrious Kirk Bauer then tweaked it a bit and twisted it into an HTTP example, which is what I'm posting below.

```
when HTTP_REQUEST timing on {
    set request_limit_reached [ table lookup "request_limit_reached" ]
    if { [expr [table incr "counter_all_requests"] % 15] == 0 } {
        if { $request_limit_reached < 2 } {
            set request_limit_reached [ table incr
"request_limit_reached" ]
            table timeout "request_limit_reached" 1
            table lifetime "request_limit_reached" 1
        }
    }
    if { $request_limit_reached >= 2 } {
        HTTP::respond 500
    }
}
```

There we go, another three killer iRules in less than 21 lines. Thanks as always to the community for doing that thing you do. Keep on rockin', and posting those iRules. If you have any questions or comments, never hesitate to drop me a line.

#Colin