Non-English DNS Request filtering via iRules

Colin Walker, 2008-20-05

When talking about iRules there are many themes that come up: application fluency, flexibility, control, high-availability. Among the themes that are often related to iRules one of the most important, in my estimation, is the extremely granular control that all of those above themes, combined with the ability to write full-fledged code against the traffic on your network, give you an amazingly granular control over the user's experience. To me, this is the "high-level" way of thinking about what iRules gives you.

Whether you're looking at custom routing L4 connections based on packet information, or sending custom built web-pages stored in a class file, complete with images, with no web-server needed, iRules can get you there. The ability to see the entire traffic flow and to act on many parts of it gives an amazing ability to understand and, if desired, alter the experience a given user is having real-time.

In the below iRule this concept is displayed in a very useful way. As the internet grows and becomes more prominent in everyday business, the world becomes much more connected. It's not uncommon these days for companies to rapidly become a global presence, even if they may have only operated locally just a few years ago. With this expansion of presence comes a whole new set of requirements to keep up with the traffic coming from all over the globe. Not the least amongst these is multi-language support.

There are services that will help translate websites, and even some software that does a halfway decent job. What about more core functionality, though? What about things like DNS requests made in a non English language? Deciphering which inbound DNS requests are English based and which aren't then routing them to the appropriate set of web-servers can be quite a task. That, however, is exactly what this iRule does quite nicely.

The below would be applied to a VIP handling DNS requests. It reads through the QNAME section of the incoming requests character by character and searches for those characters with an ASCII value higher than 127 (according to ASCII, printable English character are number 127 or lower). If it finds any, it sends the request to a pool that supports Chinese name resolution. If not, the traffic passes on its merry way to the English supporting systems.

This idea looks something like this, in code:

```
when CLIENT_ACCEPTED {
   binary scan [string range [UDP::payload] 2 3] S sflags
   set rcode  {expr $sflags & 0x000f}
   if {$rcode == 0  }{
      # skip the DNS header, jump to the QNAME part of first QUESTION
      # byte contains the first part length
      binary scan [string range [UDP::payload] 12 13 ] c foo
      # make the byte an unsigned integer
      set byte {expr $foo & 0xff}
      # initialize our posisition in the QNAME parsing and the text QNAME
      set offset 12
      # $i is a sanity check so this logic won't spin on invalid QNAMEs
      set i 0
      ############# /extract QNAME from QUESTION header #############
      while {$byte > 0 && $i < 10} {
         # grab a part and put it in our text QNAME section
         set offset {expr $offset + $byte + 1}
         # grab the length of the next part, and make it an unsigned integer
         set byte {string range [UDP::payload] $offset [expr $offset + 1]}
         binary scan $byte c foo
         set byte {expr $foo & 0xff}
         incr i
      }
      }
     #get the full domain name start and stop
     incr offset
     set i 12
     #check each charactor one by one, see if it's bigger than 127
     while {$i < $offset} {
       set byte {string range [UDP::payload] $i [expr $i+1]}
       binary scan $byte c foo
       set byte {expr $foo & 0xff}
       if {{expr $byte} >127 } {
        #log "send to Chinses dns Pool"
        pool chinese_pool
        return
        #if one char is bigger than 127, then end the rule execution
       }
       incr i
     }
    pool eng_pool
    #log "send to ENG dns"

   }
}
```

As if this cool, useful iRule weren't enough on its own, what's more is it was submitted by the community. You can check out the CodeShare entry here for the original description given by James Yang, the author of this fine bit of iRules foo. Once again, this is a fantastic display of the flexibility, application fluency, and power that is iRules.

Get the Flash Player to see this player.