Specify global variables to be used over multiple connections

Colin Walker, 2005-10-08

Keeping track of global variables is something that can be very useful when attempting to find totals and monitor maximum connections. Luckily, it’s easy to implement in an iRule.

A TCL global variable is a great way to perform this A global variable is treated the same as any other variable that you might set in your rule, except that it is using a different namespace, which allows it to be carried across multiple instances of your rule. The variable will look different because it will be pre-pended with two colons. (::variable). This is to show that it is, in fact, a global variable.

Following this approach can be useful for many applications. This example shows us how we could watch for connections identified by the client IP and reject connections over a given number, as well as log those occurrences. You can see that it initiates the array “::active_clients” when the rule starts, before any connections are made. It then tracks each connection, rejecting the connection if the incoming IP already has too many connections (>10), or incrementing the variable for the current connecting IP and allowing the connection through.

The other important part of this rule is the “CLIENT_CLOSED” section, where each closed connection is read to see if we’ve stored a variable for the IP that’s closing the connection. If we have stored a variable for this IP, the rule decrements that variable to reflect the connection that is no longer active. If this is the last connection from a given IP, the rule will unset the variable for that IP as it is no longer needed.

```
rule session_limit {
  when RULE_INIT {
    array set ::active_clients { }
  }
  when CLIENT_ACCEPTED {
    set client_ip [IP::remote_addr]
    if { [info exists $::active_clients($client_ip)] and $::active_clients($client_ip) > 10 } {
      log "Client $client_ip has too many connections"
      reject
      event CLIENT_CLOSED disable
      return
    }
    incr ::active_clients($client_ip)
  }
  when CLIENT_CLOSED {
    if { [info exists ::active_clients($client_ip)] } {
      incr ::active_clients($client_ip) -1
      if { $::active_clients($client_ip) <= 0 } {
        unset ::active_clients($client_ip)
      }
    }
  }
}
```

One thing to be conscious of when using global variables in this way, though, is that the Tcl global variables will persist indefinitely (until TMM is restarted). Therefore, these can grow to large sizes if they are not monitored and managed appropriately. It's good to use some caution when enabling variables of this kind, and only use them when actually needed.

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