Beyond HTTP - In Depth Traffic Analysis via iRules

Colin Walker, 2008-07-04

Beyond HTTP - In Depth Traffic Analysis via iRules

In today's world of increasing network traffic across just about all forms of media it becomes increasingly important to be able to analyze what's flowing across your network. This is important not only to give an accurate picture of what you are serving/requesting, but also for break/fix analysis among many other things. Maybe you're looking to help build efficiencies into your network to save costs. Maybe you're trying to chart how many requests or responses are going to/from certain locations. Maybe you're trying to troubleshoot a sticky networking issue that's causing you grief. Whatever the reason, you want information about what's on the wire in your network. This means that you need to capture information as it's being passed in order to perform analysis on that data. This iRule does exactly that.

In order, from the client being accepted at the BIG-IP, to the server connecting after the load balancing decision takes place, all the way through to the response from the Web server, this iRule paints a picture of who is requesting what, from where, and more. An organized, logical map of what's passing through your network is a hard thing to come by, which makes this example all the more powerful and useful. By keeping things organized into different Events, the log entries generated by this iRule are easy to read, logically ordered and make troubleshooting, among other things, much, much easier.

This iRule goes deeper than the often discussed HTTP or layer 7 functionality that iRules are likely most well known for. Sure, it adds in that data as well, but it's not limited to telling you which Host or URI was requested at what time. This example goes to the next step and traps information such as IP addresses, TCP ports, Mac Addresses and much, much more. By logging this information directly onto the BIG-IP, this iRule gives you a more complete picture of what is actually passing over the network, where it's coming from, where it's going, and how it's getting there.

Once this information is logged on the BIG-IP, it's really up to you how you want to make use of it. You could write an iControl application to pull the data and organize it for entry into a database. You could set up an SNMP trap to fire this info periodically off to a systems admin. Perhaps you're troubleshooting that tough networking config issue and you're just reading the logs raw. In any case, this type of in-depth information is extremely valuable.

See below for a pre-formatted, ready to run iRule that will log a huge amount of pertinent information about the connection, from beginning to end, to give you the picture you're looking for.

```
 when CLIENT_ACCEPTED {
  set info "client { [IP::client_addr]:[TCP::client_port] -> [IP::local_addr]:[TCP::local_port] }"
  append info " ethernet { [string range [LINK::lasthop] 0 16] -> [string range [LINK::nexthop] 0 16]
  log local0. $info
}

when LB_SELECTED {
  set info "client { [IP::client_addr]:[TCP::client_port] -> [clientside {IP::local_addr}]:[clientsid
  catch { append info " server { [IP::local_addr]:[TCP::local_port] -> [IP::server_addr]:[TCP::server
  append info " ethernet { [string range [LINK::lasthop] 0 16] -> [string range [LINK::nexthop] 0 16]
  log local0. $info
}

when SERVER_CONNECTED {
  set info "client { [IP::client_addr]:[TCP::client_port] -> [clientside {IP::local_addr}]:[clientsid
  append info " server { [IP::local_addr]:[TCP::local_port] -> [IP::server_addr]:[TCP::server_port] }
  append info " ethernet { [string range [LINK::lasthop] 0 16] -> [string range [LINK::nexthop] 0 16]
  log local0. $info
}

when HTTP_REQUEST {
```

```
when HTTP_REQUEST {
  set info "client { [IP::client_addr]:[TCP::client_port] -> [clientside {IP::local_addr}]:[clientsid
  catch { append info " server { [serverside {IP::local_addr}]:[serverside {TCP::local_port}] -> [IP:
  append info " ethernet { [string range [LINK::lasthop] 0 16] -> [string range [LINK::nexthop] 0 16]

  append info " - [HTTP::method] [HTTP::uri] [HTTP::version]"
  append info " *TCP MSS [TCP::mss], BW [TCP::bandwidth], RTT [TCP::rtt], OFFSET [TCP::offset]"
  append info " *IP TOS [IP::tos], HOPS [IP::hops], TTL [IP::ttl], PKTS_IN [IP::stats pkts in], PKTS_
  append info " *HTTP HOST [HTTP::host], KEEPALIVE [HTTP::is_keepalive], REQ_NUM [HTTP::request_num]"
  append info " *HTTP PATH [HTTP::path], QUERY [HTTP::query]"
  log local0. $info
}

when HTTP_RESPONSE {
  set info "client { [IP::client_addr]:[TCP::client_port] -> [clientside {IP::local_addr}]:[clientsid
  append info " server { [IP::local_addr]:[TCP::local_port] -> [IP::server_addr]:[TCP::server_port] }
  append info " ethernet { [string range [LINK::lasthop] 0 16] -> [string range [LINK::nexthop] 0 16]
  append info " - [HTTP::status] [HTTP::version] - REDIR [HTTP::is_redirect], Content-Length [HTTP::h
  append info " *TCP MSS([TCP::mss]) BW([TCP::bandwidth]) RTT([TCP::rtt]) OFFSET([TCP::offset])"
  append info " *IP TOS [IP::tos], HOPS [IP::hops], TTL [IP::ttl], PKTS_IN [IP::stats pkts in], PKTS_
  append info " *HTTP HOST [HTTP::host], KEEPALIVE [HTTP::is_keepalive], REQ_NUM [HTTP::request_num]"
  log local0. $info
}
```

Here is an example of the output that will be sent to your /var/log/ltm log file, assuming you have the default syslog configuration. As you can see there is truly a wealth of information here, right down to the Link stats on the BIG-IP.

```
Feb  1 16:18:50 tmm tmm[28526]: Rule rule_log_requests : client { 172.16.100.1:49464 -> 172.16.100.10

Feb  1 16:18:50 tmm tmm[28526]: Rule rule_log_requests : client { 172.16.100.1:49464 -> 172.16.100.10
HOPS 0, TTL 64, PKTS_IN 3, PKTS_OUT 1, BYTES_IN 345, BYTES_OUT 78 *HTTP HOST 172.16.100.100, KEEPALIV

Feb  1 16:18:50 tmm tmm[28526]: Rule rule_log_requests : client { 172.16.100.1:49464 -> 172.16.100.10

Feb  1 16:18:50 tmm tmm[28526]: Rule rule_log_requests : client { 172.16.100.1:49464 -> 172.16.100.10

Feb  1 16:18:50 tmm tmm[28526]: Rule rule_log_requests : client { 172.16.100.1:49464 -> 172.16.100.10
 REDIR 0, Content-Length 1456, Transfer-Encoding  *TCP MSS(1460) BW(0) RTT(84) OFFSET(0) *IP TOS 0, H
```

Contributed by: Mike Lowell

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