iRules IP Comparison Considerations with IP::addr Command

Jason Rahm, 2011-01-04

Anyone utilizing IP network comparisons in iRules is probably familiar with this syntax:

```
1: if { [IP::addr [IP::client_addr]/24 equals 10.10.20.0] } {
2:   ##Do this
3: }
```

In fact, there are several methods for doing a comparison. Here are three functional equivalents that include the most common form shown above:

```
[IP::addr [IP::remote_addr]/24 equals 10.10.20.0]
[IP::addr [IP::remote_addr]/255.255.255.0 equals 10.10.20.0]
[IP::addr "[IP::remote_addr] mask 255.255.255.0" equals 10.10.20.0]
```

All three work, returning true if there is match and false if not. These formats, however, are not as ideal as it was never intended to work this way. What occurs when performing the comparison this way is the system has to convert the internal IP address to string form, apply the network mask, then re-convert the result back into an IP network object for comparison to the last argument. While possible, it isn't as efficient and technically is an oversight in the syntax validation checking. It's not slated to be "fixed" at this point, but it could be in the future, so you should consider updating any iRules to one of these formats:

```
[IP::addr [IP::remote_addr] equals 10.10.20.0/24]
[IP::addr [IP::remote_addr] equals 10.10.20.0/255.255.255.0]
[IP::addr [IP::remote_addr] equals "10.10.20.0 mask 255.255.255.0"]
```

In these formats, the input address does not need to be masked and can be directly compared with the pre-parsed network specification. Finally, there is one more format that works:

```
[IP::addr [IP::addr [IP::remote_addr] mask 255.255.255.0] equals 10.10.20.0]
```

This also doesn't require any additional conversion, but the sheer volume of commands in that statement is an immediate indicator (to me, anyway) that it's probably not very efficient.

Performance

Before running each format through a simple test (ab –n 10000 –c 25 http://<vip>), I set up a control so I could isolate the cycles for each format:

```
when HTTP_REQUEST timing on {
### CONTROL ###
  if { 1 } { }
}
```

Since all the formats will return true if the match is found, then subtracting out the average cycles from this iRule should give me a pretty accurate accounting of cycles required for each specific format. So I can replace the "1" from the conditional with each format, as shown in this iRule:

```
when HTTP_REQUEST timing on {
### FUNCTIONAL EQUIVALENTS "RECOMMENDED" ###
#     Format #1 Cycles: 6839 - 1136 = 5703
#     if { [IP::addr [IP::remote_addr] equals 10.10.20.0/24] } { }
#     Format #2 Cycles: 6903 - 1136 = 5767
#     if { [IP::addr [IP::remote_addr] equals 10.10.20.0/255.255.255.0] } { }
```

```
#     Format #3 Cycles: 7290 - 1136 = 6154
#     if { [IP::addr [IP::remote_addr] equals "10.10.20.0 mask 255.255.255.0"] } { }

### FUNCTIONAL EQUIVALENTS "NOT RECOMMENDED" ###
#     Format #4 Cycles: 8500 - 1136 = 7364
#     if { [IP::addr [IP::remote_addr]/24 equals 10.10.20.0] } { }
#     Format #5 Cycles: 8543 - 1136 = 7407
#     if { [IP::addr [IP::remote_addr]/255.255.255.0 equals 10.10.20.0] } { }
#     Format #6 Cycles: 8827 - 1136 = 7691
#     if { [IP::addr "[IP::remote_addr] mask 255.255.255.0" equals 10.10.20.0] } { }

### ALTERNATE FORMAT ###
#     Format #7 Cycles: 9124 - 1136 = 7988
#     if { [IP::addr [IP::addr [IP::remote_addr] mask 255.255.255.0] equals 10.10.20.0]    } { }

### CONTROL ###
# Cycles: 1136
#  if { 1 } { }
}
```

You can see from the average cycles data I added to each of the formats comments above that the recommended formats are all faster than the formats that are not recommended. What's interesting is the subtle differences in the "/" formats within each group of functional equivalents and then the significant outliers that the "<net> mask <mask>" formats are within their group. Also of note is that the last format, while acceptable, is really inefficient and should probably be avoided. The table below breaks down the increase in cycles for each of the formats compared to the best performer: [IP::addr [IP::remote_addr] equals 10.10.20.0/24].

Comparisons to **[IP::addr [IP::remote_addr] equals 10.10.20.0/24]**	
Command Format	Increase in Cycles
[IP::addr [IP::remote_addr] equals 10.10.20.0/255.255.255.0]	1.1%
[IP::addr [IP::remote_addr] equals "10.10.20.0 mask 255.255.255.0"]	7.9%
[IP::addr [IP::remote_addr]/24 equals 10.10.20.0]	29.1%
[IP::addr [IP::remote_addr]/255.255.255.0 equals 10.10.20.0]	29.9%
[IP::addr "[IP::remote_addr] mask 255.255.255.0" equals 10.10.20.0]	35.0%
[IP::addr [IP::addr [IP::remote_addr] mask 255.255.255.0] equals 10.10.20.0]	40.1%

Whereas the number of cycles required to execute this operation are all quite small, the difference beyond the first two similar formats is quite significant.

Conclusion

Many ways to skin a cat, some good, some not so good. Just to be clear, using the mask on the initial argument of the comparison should be avoided, and if you currently have iRules utilizing this format, it would be best to update them sooner rather than later. I'll be doing the same on all the DevCentral wikis, forums, articles, and blogs. If you find some references where we haven't made this distinction, please comment below.

Related Articles

- IP::addr and IPv6
- DevCentral Wiki: IP::addr
- Why do we need to use [IP::addr [IP::client_addr]]? - DevCentral ...
- IP Address based iRule - DevCentral - F5 DevCentral > Community ...

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