Specifying which headers to allow with an HTTP request

Colin Walker, 2005-10-08

Sometimes, in the name of security, you'd just as soon not allow every single header type through to your web servers without inspecting the information they contain. Well, lucky for you there's a way to weed out the bad seeds, so to speak, with a relatively simple iRule that can work wonders for the security minded, or even just curious.

The rule would look something like this:

```
set http_headers [HTTP::header names]
for { } { 1 } { } {
  set index [lsearch $http_headers "badHeader"]
  if {$index != -1} {
    set http_headers [lreplace $http_headers $index $index ]
  } else {
    break
  }
}


HTTP::header sanitize $http_headers
```

The first thing that this rule does is set up a variable to represent the HTTP headers. It then searches through that string to find whatever headers you've determined you don't want to be passed on to the webserver. This is the string "badHeader" in the example above.

If it finds the "badHeader" string, it will remove it from the http_headers string. Once the iRule has searched for the headers and the http_headers string is properly re-formatted (minus any of the unwanted header names), it invokes the HTTP::header sanitize command, passing it the newly cleansed version of the http_headers string. This should effectively remove all of those bad header fields, and their corresponding data, from the HTTP header before ever allowing it off of the BigIP.

With this iRule, you can offload at least a portion of the security responsibilities from your webservers, which are likely patched, upgraded and expanded frequently. An advantage to using an iRule like this is that it means one less thing for you to keep track of on the server side.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com