Validate String Characters in Cookie

Use Case:

HTTP Cookie are key to Application access and persistence and post authentication, Cookie becomes the way to let access go through as a trusted channel. Certain Apps define what they do not expect in the Cookie value and ADC can help enforce the check before the request makes it to backend. In this example when HTTP request contains cookie header, we check whether value of the cookie contains any characters not defined in the configured legal list of characters and log that entry to Audit Log.

F5 iRules:

```
when RULE_INIT {


   # Set the name of the cookie to validate

   set ::cookie_to_validate "my_cookie"


   # Log debug messages to /var/log/ltm?  1=yes, 0=no.

   set ::cookie_validation_debug 1


   # Character set validation:

   # The format is {%[CHARS]}, where CHARS can be a character ranges or single ch
aracters.

   # For details on configuring the characters, refer to the TCL man page for 'sc
an'

   # The literal hyphern character '-' needs to be listed first or last in the ch
aracter set

   set ::allowed_chars_cookie_value {%[-a-zA-Z0-9_]}

}


when HTTP_REQUEST {


   # Check if the cookie is present in a request and has a length
```

```
   if {[HTTP::cookie value $::cookie_to_validate] ne ""}{


      # Check if the cookie value contains any illegal characters

      if {[HTTP::cookie value $::cookie_to_validate] eq [scan [HTTP::cookie value
$::cookie_to_validate] $::allowed_chars_cookie_value]}{

         # Cookie contains only valid characters

         # Log a message if debug is enabled

         if {$::cookie_validation_debug}{log local0. "[IP::client_addr]:[TCP::cli
ent_port]: Request with legal cookie value: [HTTP::cookie value $::cookie_to_vali
date]"}


      } else {


         # Cookie contained invalid characters


         # Log a message if debug is enabled

         if {$::cookie_validation_debug}{


         set len [string length [scan [HTTP::cookie value $::cookie_to_validate]
$::allowed_chars_cookie_value]]


         log local0. "[IP::client_addr]:[TCP::client_port]: Request with illegal
cookie value: [HTTP::cookie value $::cookie_to_validate], \

            char: [string range [HTTP::cookie value $::cookie_to_validate] $len $
len]"}

      }

   }

}
```

NetScaler Solution:

add audit messageaction cookie_invalid_act INFO "\"Client IP: \" + CLIENT.IP.SRC + \"Client Port Number is: \"+CLIENT.TCP.SRCPORT + \"Request with Illegal cookie value\" + HTTP.REQ.HEADER(\"Cookie\"). TYPECAST_NVLIST_T('=',';').VALUE(0)" -by yes -logtoNewnslog YES

add rewrite policy test_cookie_fail 'HTTP.REQ.HEADER(\"Cookie\"). TYPECAST_NVLIST_T('=',';').VALUE(0).REGEX_MATCH(re/^[-0-9A-Za-z_]*$/).NOT' NOREWRITE -logAction cookie_invalid_act

add audit messageaction cookie_valid_act INFO "\"Client IP: \" + CLIENT.IP.SRC + \"Client Port Number is: \"+CLIENT.TCP.SRCPORT + \"Request with legal cookie value\" + HTTP.REQ.HEADER(\"Cookie\").TYPECAST_NVLIST_T('=',';').VALUE(0) " -by yes -logtoNewnslog YES

add rewrite policy test_cookie_pass 'HTTP.REQ.HEADER(\"Cookie\") TYPECAST_NVLIST_T('=',';').VALUE(0).REGEX_MATCH(re/^[-0-9A-Za-z_]*$/)' NOREWRITE -logAction cookie_valid_act

Bind rewrite policy to specific VSERVER or to Global rewrite bind point on Request flow.

Here we are validating only for the first cookie value coming in the request payload. Customer can choose which cookie value to validate if there are more Cookie headers in the request.