Redirect Location header validator

Use Case:

Location header in HTTP responses are used to redirect client to different source. Most of the HTTP Redirection happens based on Location header processing. In this use case Location header is intercepted for any redirects which are to a domain that is not in a whitelist string data group. If it is not in whitelist, Location header will change to default blocked HTML page.

F5 iRules:

```
# Validate the 30x redirects the web application sends to clients against a white list stored in a data group

# The goal is to prevent the client from being tricked into receiving an unvalidated redirect to a malicious third party site

# by an attacker.  See the OWASP Top Ten for details on the vulnerability: https://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards


# The data group format is string.  v11 example:

#

#ltm data-group internal my_domain_whitelist_dg {

#    records {

#        example.com { }

#        example.net { }

#        example.org { }

#        example.co.uk { }

#        partner.com { }

#    }

#    type string

#}


when RULE_INIT {

   # Name of the string data group which contains the whitelisted domains

   # (ex: legal.com, legal.net, okay.org, etc)
```

```
   set static::domain_whitelist_dg "my_domain_whitelist_dg"


   # Log debug to /var/log/ltm? 1=yes, 0=no.
   set static::redirect_debug 1
}
when HTTP_RESPONSE {


   # Check if domain is a redirect
   if {[HTTP::is_redirect]}{


      if {$static::redirect_debug}{
         log local0. "[IP::client_addr]:[TCP::client_port]: status=[HTTP::status]
, Location=[HTTP::header Location], Server=[LB::server]"
      }
      # Check if the redirect location is an absolute URL
      switch -glob [HTTP::header Location] {
         "/*" -
         "" {
            # No Location header value or a local reference so do not check again
st whitelist
            if {$static::redirect_debug}{log local0. "[IP::client_addr]:[TCP::cli
ent_port]: Location local or empty. Not validating"}
         }
         default {
            # Check Location value against the data group
            set hostname [string tolower [URI::host [HTTP::header Location]]]
            if {$static::redirect_debug}{log local0. "[IP::client_addr]:[TCP::cli
ent_port]: Checking $hostname against $static::domain_whitelist_dg"}


            # Check the hostname against the whitelist
            if {[class match $hostname ends_with $static::domain_whitelist_dg]}{
               if {$static::redirect_debug}{log local0. "[IP::client_addr]:[TCP::
client_port]: Legal redirect to [HTTP::header location]"}
```

```
            } else {
               if {$static::redirect_debug}{log local0. "[IP::client_addr]:[TCP::
client_port]: Illegal redirect to [HTTP::header location]"}
               HTTP::respond 302 Location "http://www.example.com/redirect_blocke
d.html"
            }
         }
      }
   }
}
```

NetScaler Solution:

add patset records
bind patset records example.com
bind patset records example.net
bind patset records example.urg
bind patset records allowed.com
bind patset records partner.com

add rewrite action block_list_act replace HTTP.RES.HEADER("Location")
'"http://www.example.com/redirect_blocked.html"'

add rewrite policy pol111
'HTTP.RES.HEADER("Location").TYPECAST_HTTP_URL_T.CONTAINS_ANY("records").
NOT' block_list_act

Here NetScaler is evaluating the Location header value for the Whitelist created by patset configuration. In case it does not match to any of the patset entry, will be redirected to the fixed URL.