iRule Security 101 - #02 - HTTP Methods and Cross Site Tracing

Joe Pruitt, 2007-13-08

In this installment of iRule Security 101, I'll refer to OWASP's Data Validation Test "Testing for HTTP Methods and XST (Cross Site Tracing)" and illustrate how to use iRules to build a defense mechanism to block potentially dangerous HTTP commands (methods) and ensure that Cross Site Tracing is not possible. Other articles in the series:

- iRule Security 101 – #1 – HTTP Version
- iRule Security 101 – #02 – HTTP Methods and Cross Site Tracing
- iRule Security 101 – #03 – HTML Comments
- iRule Security 101 – #04 – Masking Application Platform
- iRule Security 101 – #05 – Avoiding Path Traversal
- iRule Security 101 – #06 – HTTP Referer
- iRule Security 101 – #07 – FTP Proxy
- iRule Security 101 – #08 – Limiting POST Data
- iRule Security 101 – #09 – Command Execution

GET and POST methods the most used methods to request information from a web server, but the HTTP protocol allows several others including HEAD, PUT, DELETE, TRACE, OPTIONS, and CONNECT. Some of these can cause potential security risks such as remote file access and un-intended retrieval of information. The TRACE in particular, was designed to allow echoing of strings sent to the server and is mainly used for debugging purposes. But, the use of this method has been shown to allow for an attach known as Cross Site Tracing, which was discovered by Jeremiah Grossman in his paper titled "Cross Site Tracing (XST)".

```
when RULE_INIT {
  set INFO 0
  set DEBUG 0
  #--------------------------------------------------------------------------
  # HTTP Method
  #--------------------------------------------------------------------------
  set sec_http_method_enabled 1
  set sec_http_method_block 1
  set sec_http_method_alert 1
  set sec_http_methods [list \
    "CONNECT" \
    "DELETE" \
    "HEAD" \
    "OPTIONS" \
    "PUT" \
    "TRACE" \
  ]
}

when HTTP_REQUEST {
  #==========================================================================
  # HTTP Method
  #==========================================================================
  if { $::INFO } { log local0. "ASSERTION: http_method" }
  if { $::sec_http_method_enabled } {
    if { $::DEBUG } { log local0. "  HTTP Method: [HTTP::method]" }
    if { [matchclass [HTTP::method] equals $::sec_http_methods] } {
      if { $::sec_http_method_alert } { log local0. "  SEC-ALERT: Invalid HTTP Version found: '[HTTP:
```

```
      if { $::sec_http_method_block } { reject }
    } else {
      if { $::DEBUG } { log local0. "  PASSED" }
    }
  }
}
```

In the RULE_INIT method, I've created a few global variables enabling one to turn on or off the verification. This iRule will block all requests in the violation list so make sure before you deploy this iRule that you verify that your web applications don't make use of any of them.

Without all the extra conditionals, the iRule can be stripped down to the following couple of lines:

```
when RULE_INIT {
  set sec_http_methods [list "CONNECT" "DELETE" "HEAD" "OPTIONS" "PUT" "TRACE"]
}

when HTTP_REQUEST {
  if { [matchclass [HTTP::method] equals $::sec_http_methods] } {
    reject
  }
}
```

For more information on HTTP Method attacks and Cross Site Tracing, take a look at OWASP's documentation on the topic.

Get the Flash Player to see this player.

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