**Http Request Tampering**

**Description 1**

**Other Name: Http Data Tampering**

Most of the time you are using the HTTP protocol (or the HTTPS protocol) when you want to watch websites. Browsers (Firefox, Chrome, etc) are applications that use this protocol to communicate with web servers (apache, IIS, etc.)  and interpret HTML, CSS with JavaScript.

Whenever you type a URL, click on a link, send a form, your browser send an HTTP request with different options to the web server that host the website. When the web server get the HTTP request, it communicates with the backend systems (database, php, etc) and forges an HTTP response header together with the file requested, most of the time an HTML page. Once created, it sends the file to you. Your browser interpreted the page and render it. HTTP request doesn’t have to be built from a browser, it’s possible to forge your own one. Tampering data, in HTTP communication, is the act of modifying data of the HTTP request (or response) before the recipient read it. [1]

Note: According to [1] The HTTP request header contains several options but the most interesting is Cookie. Therefore, the consequences of Http Request Tampering are mainly similar to cookie tampering (cookie poisoning)

**Description 2**

**Other Name: Http verb Tampering**

Hypertext transfer protocol (HTTP) gives you list of methods that can be used to perform actions on the web server. In HTTP methods, GET and POST are most commonly used by developers to access information provided by a web server. HTTP allows several other methods as well, which are less known methods. Following are some of the methods:

* **HEAD**
* **GET**
* **POST**
* **PUT**
* **DELETE**
* **TRACE**
* **OPTIONS**
* **CONNECT**

Many of these methods can potentially pose a critical security risk for a web application, as they allow an attacker **to modify the files stored on the web server, delete the web page on the server, and upload a web shell to the server which leads to stealing the credentials of legitimate users**. Moreover, when rooting the server, the methods that must be disabled are the following:

* **PUT**: This method allows a client to upload new files on the web server. An attacker can exploit it by uploading malicious files (e.g. an ASP or PHP file that executes commands by invoking cmd.exe), or by simply using the victim’s server as a file repository.
* **DELETE**: This method allows a client to delete a file on the web server. An attacker can exploit it as a very simple and direct way to deface a web site or to mount a Denial of Service (DOS) attack.
* **CONNECT**: This method could allow a client to use the web server as a proxy
* **TRACE**: This method simply echoes back to the client whatever string has been sent to the server, and is used mainly for debugging purposes of developers. This method, originally assumed harmless, can be used to mount an attack known as Cross Site Tracing, which has been discovered by Jeremiah Grossman.

**Description 3**

An attacker can manipulate http verb to attempt **to** **bypass or circumvent security controls**. The attacker can try one of valid http verbs, including trace, track, put and delete. Attacker can also try arbitrary strings, such as JEFF, as the HTTP web.

**Reference**

**[1]** [**https://toschprod.wordpress.com/2011/10/13/http-data-tampering/**](https://toschprod.wordpress.com/2011/10/13/http-data-tampering/)

**[2]** [**https://www.owasp.org/index.php/Testing\_for\_HTTP\_Verb\_Tampering\_(OTG-INPVAL-003)**](https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_(OTG-INPVAL-003))

**[3]** [**http://cdn2.hubspot.net/hub/315719/file-1344244110-pdf/download-files/Bypassing\_VBAAC\_with\_HTTP\_Verb\_Tampering.pdf?t=1494862850005**](http://cdn2.hubspot.net/hub/315719/file-1344244110-pdf/download-files/Bypassing_VBAAC_with_HTTP_Verb_Tampering.pdf?t=1494862850005)

**( reference [3] used by owasp)**