Logo

Description automatically generated

HTTP Method Validation/Exploitation

Contents

[What is HTTP Method Exploitation? 3](#_Toc132380765)

[Why Use HTTP Method Exploitation? 4](#_Toc132380766)

[Attempts of HTTP Method Exploitation 4](#_Toc132380767)

[Tools used: 4](#_Toc132380768)

[PUT Request 4](#_Toc132380769)

[DELETE Request 6](#_Toc132380770)

[TRACE Request 6](#_Toc132380771)

[OPTIONS Request 6](#_Toc132380772)

[CONNECT Request 7](#_Toc132380773)

[PATCH Request 7](#_Toc132380774)

[HEAD Request 8](#_Toc132380775)

[Final Findings 9](#_Toc132380776)

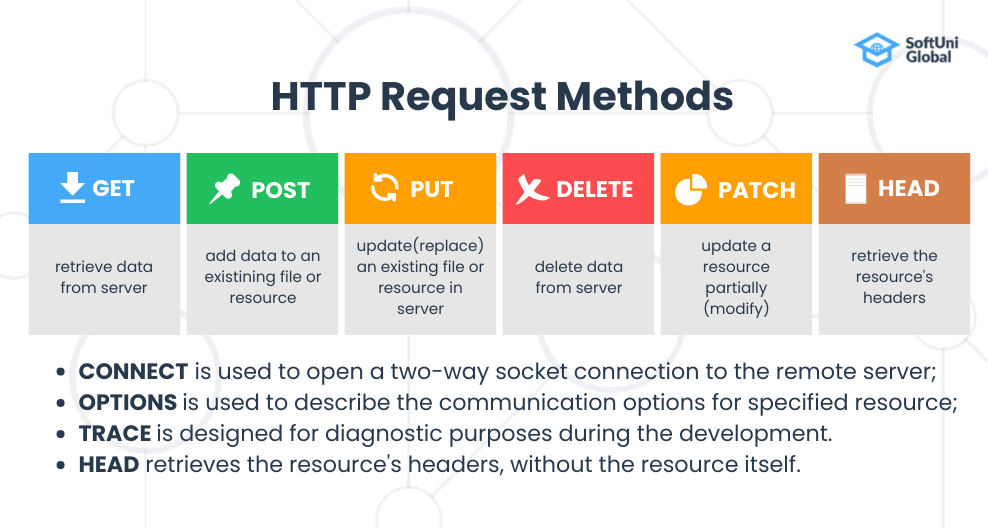
Graphical user interface, text, application

Description automatically generated

# What is HTTP Method Exploitation?

HTTP Method validation is the process of how a web app will handle different HTTP methods to ensure only correct methods can be input.

Meanwhile HTTP method exploitation is using HTTP methods maliciously to find out vulnerabilities and exploit how a web application handles the http methods. Below is an example of the different HTTP methods that are used within any application.



*Figure 1. HTTP Request Methods by SoftUni Global*

Each method has its own purpose and exploiting them can have vastly different outcomes. For example, someone with purely malicious intent can use the delete method to remove the data from the server.

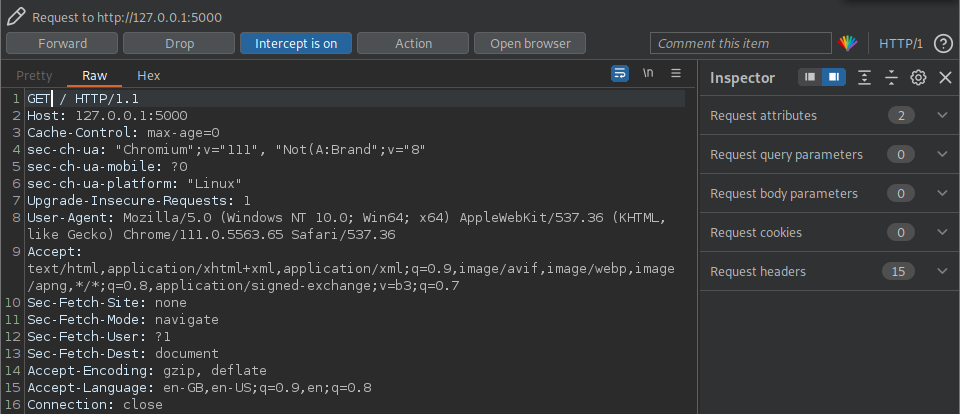
# Why Use HTTP Method Exploitation?

Using HTTP method exploitation can allow for any unauthorized person to bypass security methods and performing unauthorized actions, if a webapp is configured incorrectly it can allow almost anyone to act as an admin. Attackers using method exploitation can become a powerful tool to cause disruption to a perfectly functioning webapp and can be used to bypass authentication. If the patch method is allowed, it could allow for any attacker to upload whatever file they’d like to the attacked app and it goes without saying that could cause mayhem on any app.

# Attempts of HTTP Method Exploitation

## Tools used:

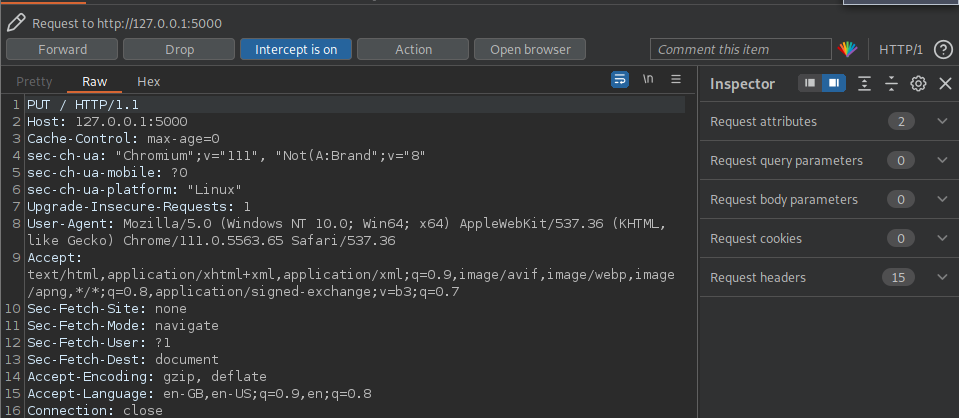
* ParrotOS – A free Debian based operating system that focuses primarily on security
* Burpsuite – a software application used for security testing. It includes many different tools and has many different modules that can be used for pentesting. During the testing I focused on intercepting the packets and modifying them to attempt HTTP method exploitation.



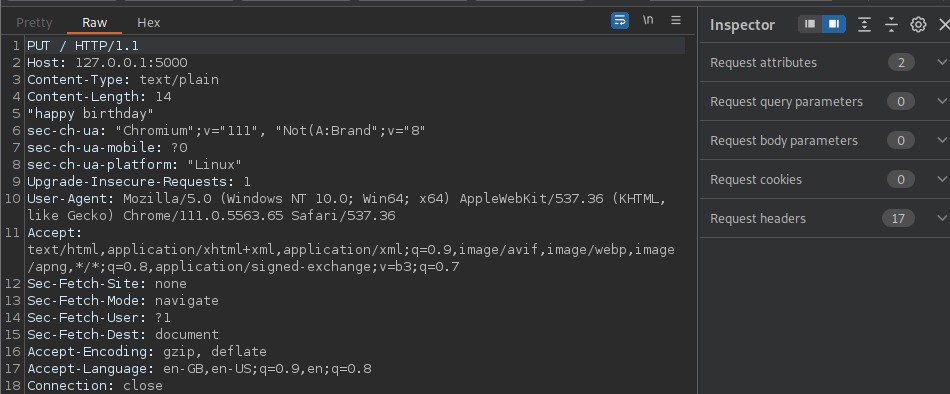
Above is the standard packet that gets intercepted, by burpsuite requesting the homepage of the webapp.

## PUT Request

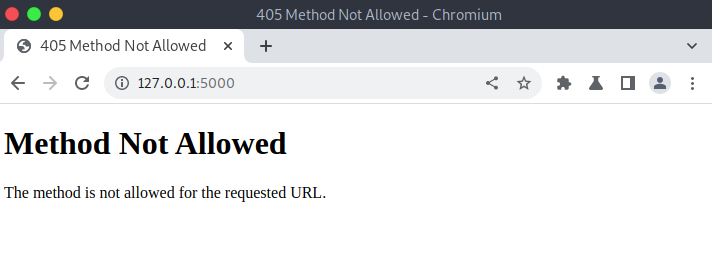
Using burpsuite allows us to not intercept the packet but to also modify it and then forward it onto the server as shown below, I have put a PUT request, which instead of just obtaining the webpage will try to update the webpage.



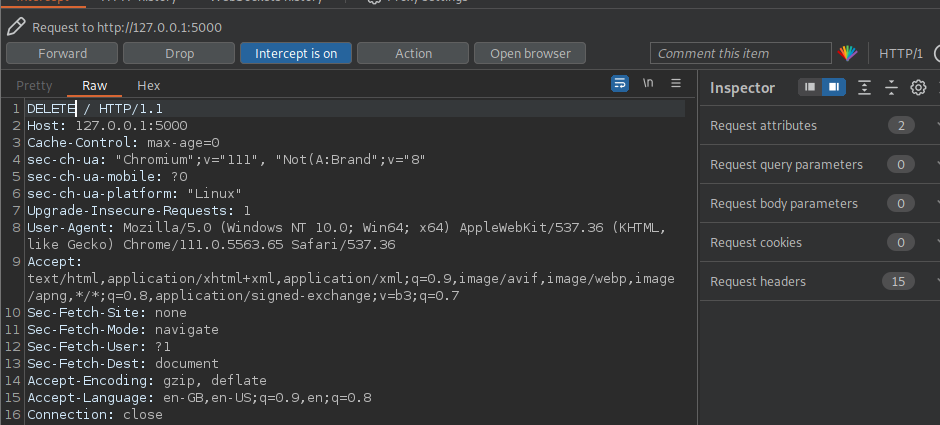
I tried different messages for the webapp such as happy birthday (below) but was unable to get any PUT requests through,



However when forwarding the packet to the server it shows that the average user cannot do put requests.



## DELETE Request



In this test I tried to delete the homepage itself, and while this would be terrible practice as it could cause the whole app to break I was once again greeted with this message.

Graphical user interface, text, application

Description automatically generated

## TRACE Request

Attackers could use the trace requests to see if validation is done on the client side or on the server side and if it is done client sided it can allow for easier vulnerabilities, since there is nowhere to login yet on this webapp there wasn’t any point of doing a Trace Request

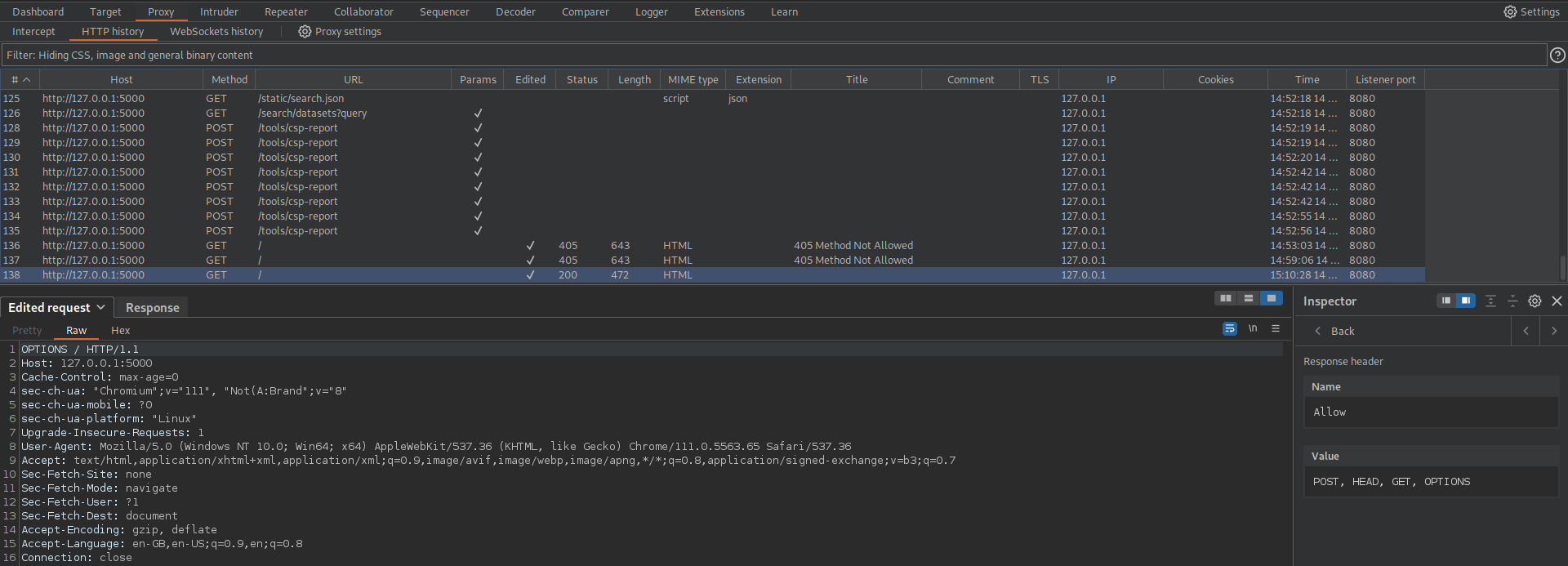
## OPTIONS Request

Ideally an options request would be the best way to start, as you can use this request to see what requests are allowed. Unlike the many other requests before this one we can see results.

Graphical user interface, application

Description automatically generated

What looks like a blank page to most people is actually the page telling us what requests are allowed, since the web app did respond with the available options in the header as shown in the burpsuite screenshot below.



Through this screenshot we can see that there are 4 requests that any user can do: GET, POST, HEAD, and OPTIONS

## CONNECT Request

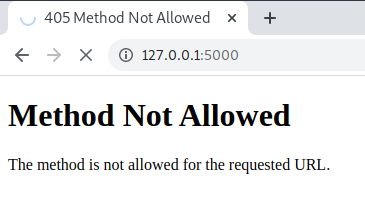
A connect Request will allow attackers to try and establish a connection via a proxy server to the webapp, however it requires SSL/TLS and thus is pointless to try that kind of request on this webapp.

## PATCH Request

Text

Description automatically generated

A Patch request will be used to update resources of the website, I tried to update the page to simply show a message but as shown above we can see what requests are allowed and because of this we were once again greeted with this page:



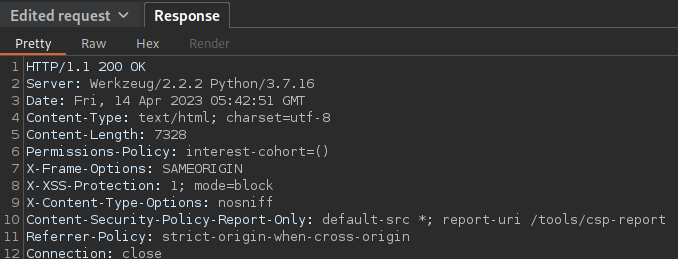
## HEAD Request

The HEAD request is similar to a get request however instead of getting a webpage itll only retrieve for the page and thus the result of the page loading for a general user will be a blank page like below.

Graphical user interface, application

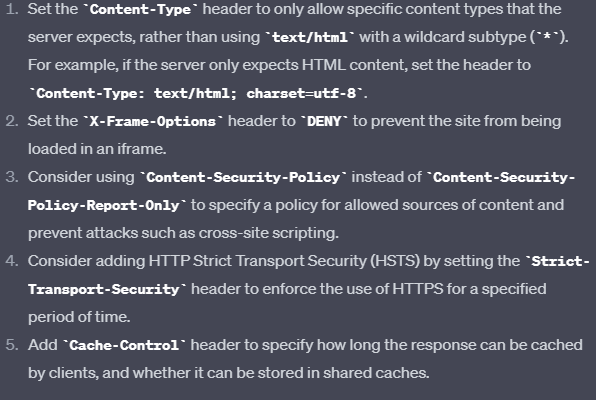
Description automatically generated

The metadata can be seen in the burpsuite response as shown below. We can see all the fields including the server information and other important information such as XSS protection. At the moment this field isn’t relevant since the forms on the web app don’t work just yet.



# Final Findings

While we can see what requests can be accessed, through the webapp, there is only one that could be a future concern, the POST request, since that can be used with XSS (Cross- Site Scripting) attacks however now the forms are unusable so would currently be unusable. The web app can become slightly more secure if OPTIONS requests were only available to authorized users and if the HEAD request were disabled, the HEAD Request provides enough information for an attacker to gain insight and prepare for an attack. Disabling it for unauthorized users would provide one extra layer of security. After running the response for the HEAD request through ChatGPT for any possible improvements it was able to come up with these potential improvements (shown in the screenshot below)



However as mentioned above these improvements don’t need to be implemented if the HEAD request is disabled for regular users as attackers wouldn’t be obtain this information to begin with but if these measures were implemented then perhaps there would be nothing left to improve.