CHAMELEON

FOR OUR SMARTER WORLD

HTTP Methods – Attack Report

Hamish Burnett – (222282244)

Contents

Executive Summary

In this investigation, the attack that will be analysed, is a HTTP Method attack. HTTP Methods are used to specify an action for the webserver to complete, when a user launches a request (OWASP Foundation, n.d.-a). Common methods include GET, POST, and HEAD, and uncommon methods include OPTIONS, PUT, and DELETE. A HTTP Method attack is an attack that forces the webserver to utilise malicious HTTP Methods, rather than safe methods such as GET and POST. These malicious methods can be used to upload malicious files to the server (PUT), and delete files on the server (DELETE).

The tools that were utilised, included Kali Linux as the Operating System, Burp Suite, to modify the methods in the requests, FoxyProxy, to redirect traffic to Burp Suite, and Curl, a command line interface tool to create requests with malicious HTTP methods. The website that was tested, was the Chameleon website.

As a result of the investigation, it was found that the website is not vulnerable to HTTP Method attacks. It was found that the webserver had been configured correctly, which prevented these attacks from being successful.

As the testing found that the website is not vulnerable to this attack, no further actions need to be taken, in regards to this matter.

Introduction

A HTTP Method attack is an attack that forces a website server to utilise malicious HTTP Methods, rather than safe methods (OWASP Foundation, n.d.-b). Methods are used in HTTP requests to perform specific actions (OWASP Foundation, n.d.-a). The most common methods, include the GET, and POST methods. GET methods are used to retrieve information from a server, and POST methods are used to send data to the server. Methods that can be used for malicious purposes include the PUT, DELETE, CONNECTION, and OPTIONS methods. The PUT method is used to upload a file to the webserver, the DELETE method is used to delete files on the webserver, the CONNECTION method is utilised to form a connection, which can be used for malicious purposes, and the OPTIONS method is used to list which HTTP methods are available.

The operation of this security test will determine whether the Chameleon website is vulnerable to HTTP Method attacks, and if so, present recommendations to prevent HTTP Method attacks in the future.

Tools Used

Four main tools were used, which are listed below:

- Kali Linux
- Burp Suite
- FoxyProxy
- Curl

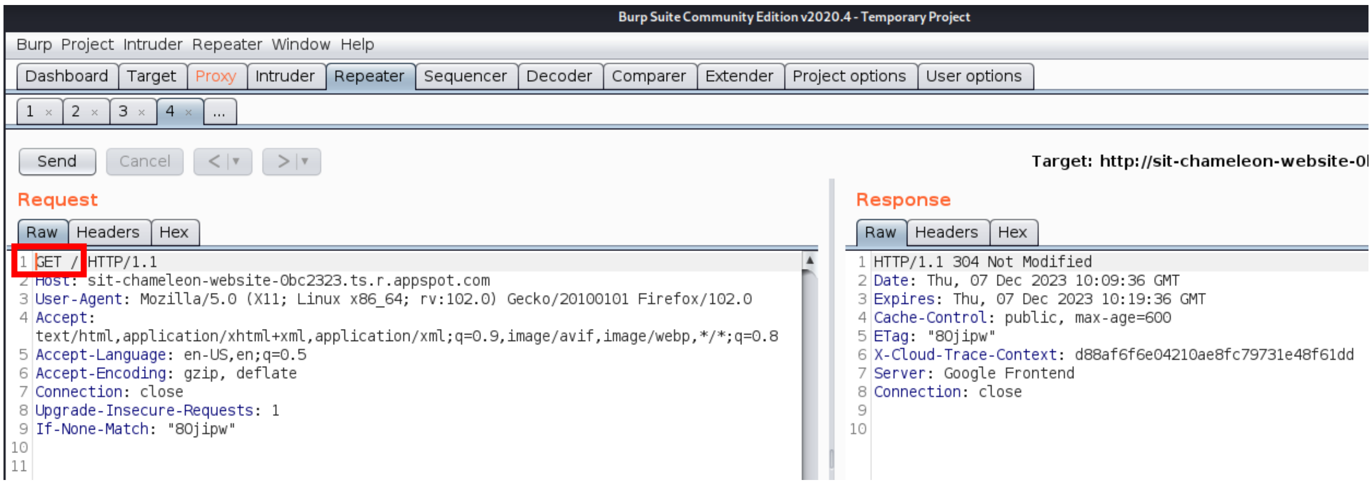
The attacks were performed using Kali Linux as the Operating System (OS), through a virtual machine. Burp Suite was used to intercept requests, and modify them, to test for HTTP Method attack. FoxyProxy was used to Burp Suite. Curl was used as a command line interface tool, to send requests to the webserver, with malicious methods.

Scope of Testing

The scope of this security test, was limited to the Chameleon website. Burp Suite and Curl were used to launch the attacks.

Methodology

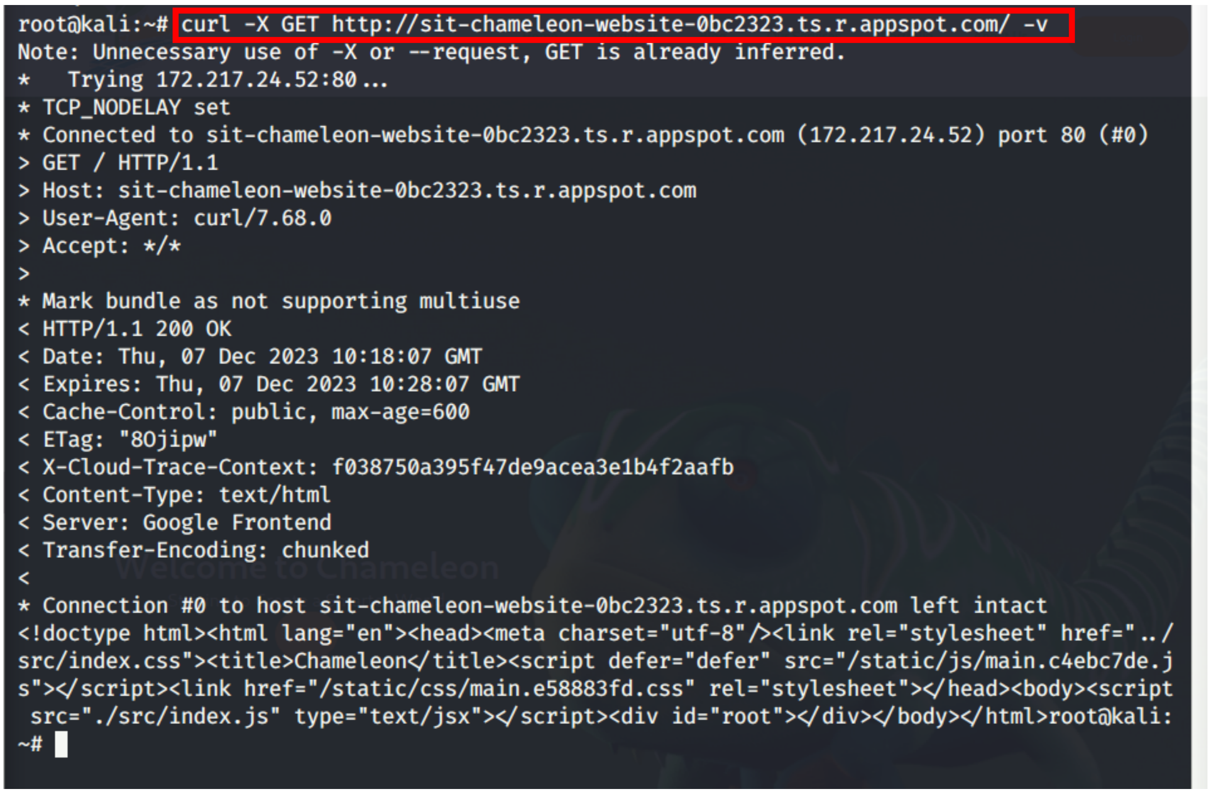
Several different HTTP Method attacks were attempted using Burp Suite, before attempting similar attacks through Curl. To use Burp Suite, FoxyProxy was activated, which redirected the traffic to Burp Suite. In Burp Suite, the traffic was sent to the repeater, which enabled multiple attack variations to be tested in a fast manner. Once a request was modified in Burp Suite, and sent to the webpage, the response was analysed. Key features that were analysed, included the HTTP Response code, and the content of the response. Figure 1 shows a screenshot of Burp Suite, with the method circled in red, which will be modified for each attack. Figure 2 shows a screenshot of a Curl command (shown in red) and the associated response.


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
1 2 3 4 ...
Send Cancel
Target: http://sit-chameleon-website-0
Request
Raw Headers Hex
Response
Raw Headers Hex

```
GET / HTTP/1.1
Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
```

```
HTTP/1.1 304 Not Modified
Date: Thu, 07 Dec 2023 10:09:36 GMT
Expires: Thu, 07 Dec 2023 10:19:36 GMT
Cache-Control: public, max-age=600
ETag: "80jipw"
X-Cloud-Trace-Context: d88af6f6e04210ae8fc79731e48f61dd
Server: Google Frontend
Connection: close
```

Figure 1: Screenshot showing Burp Suite, with the Method (GET in the Screenshot) that will be Modified for the Attacks (ircled in Red).

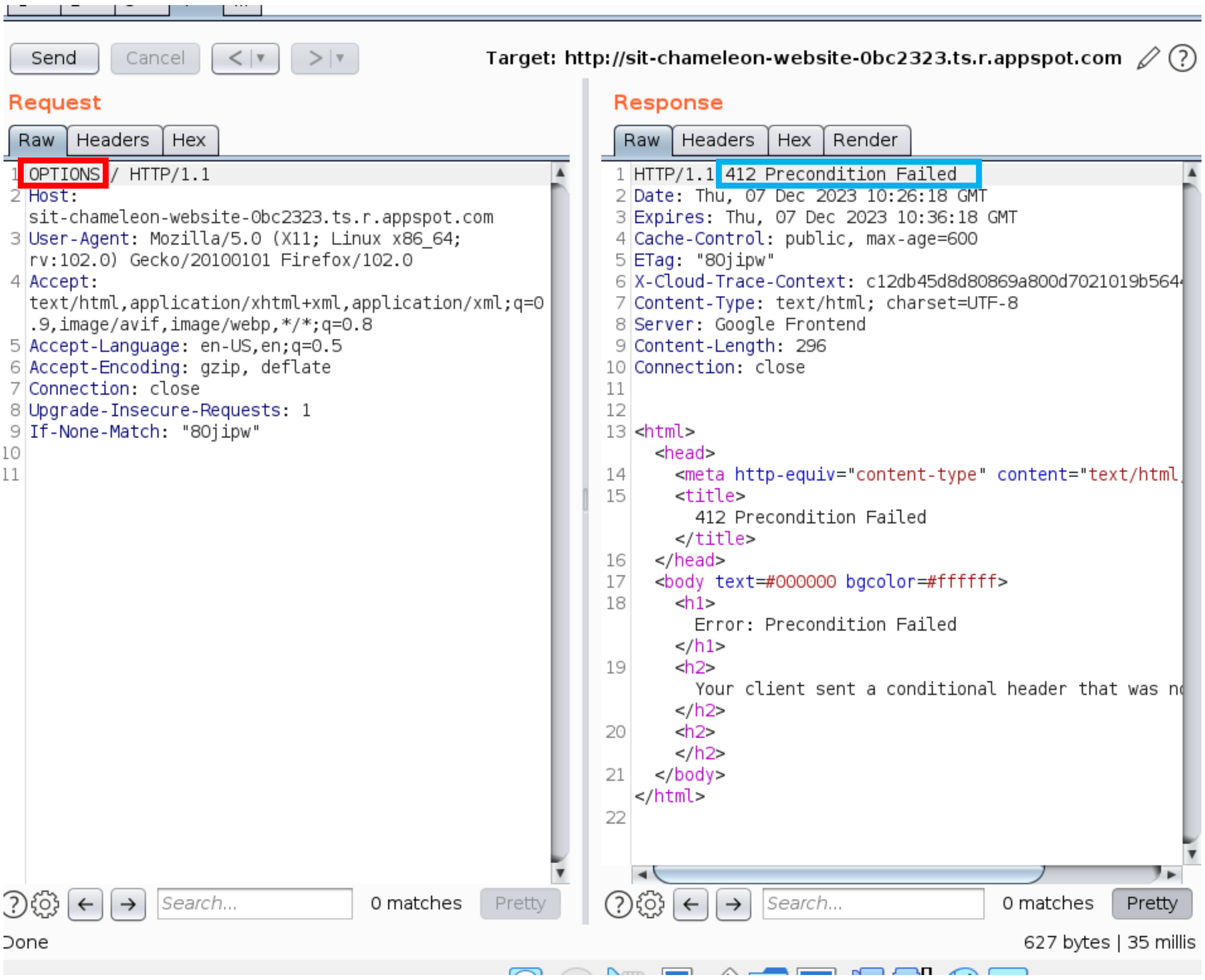


```
root@kali:~# curl -X GET http://sit-chameleon-website-0bc2323.ts.r.appspot.com/ -v
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 172.217.24.52:80...
* TCP_NODELAY set
* Connected to sit-chameleon-website-0bc2323.ts.r.appspot.com (172.217.24.52) port 80 (#0)
> GET / HTTP/1.1
> Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 07 Dec 2023 10:18:07 GMT
< Expires: Thu, 07 Dec 2023 10:28:07 GMT
< Cache-Control: public, max-age=600
< ETag: "80jipw"
< X-Cloud-Trace-Context: f038750a395f47de9acea3e1b4f2aafb
< Content-Type: text/html
< Server: Google Frontend
< Transfer-Encoding: chunked
<
* Connection #0 to host sit-chameleon-website-0bc2323.ts.r.appspot.com left intact
<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="stylesheet" href="../
src/index.css"><title>Chameleon</title><script defer="defer" src="/static/js/main.c4ebc7de.j
s"></script><link href="/static/css/main.e58883fd.css" rel="stylesheet"></head><body><script
 src="./src/index.js" type="text/jsx"></script><div id="root"></div></body></html>root@kali:
~# 
```

Figure 2: Screenshot showing Operation of Curl (Command shown in Red), to Test the Website.

Results

The HTTP Method OPTIONS was first to be tested in Burp Suite. The OPTIONS method lists all the methods (i.e. GET, POST, PUT, DELETE, HEAD, OPTIONS) that the website will accept. From there, these methods can be utilised to perform the attack. The method OPTIONS was entered into the request in Burp Suite (See red box, in Figure 3), and the result is shown on the right. It can be seen that the response code (circled in blue) is code 412, which indicates that this option has been denied by the webserver (Mozilla, 2023d). When analysing the header (lines 1-11) in the response, the methods that the website can accept are not shown. This is ideal, as it indicates that the webserver may have been hardened against this attack. It can also be seen in the content section of the response (lines 12-22) that an error is displayed to the user.

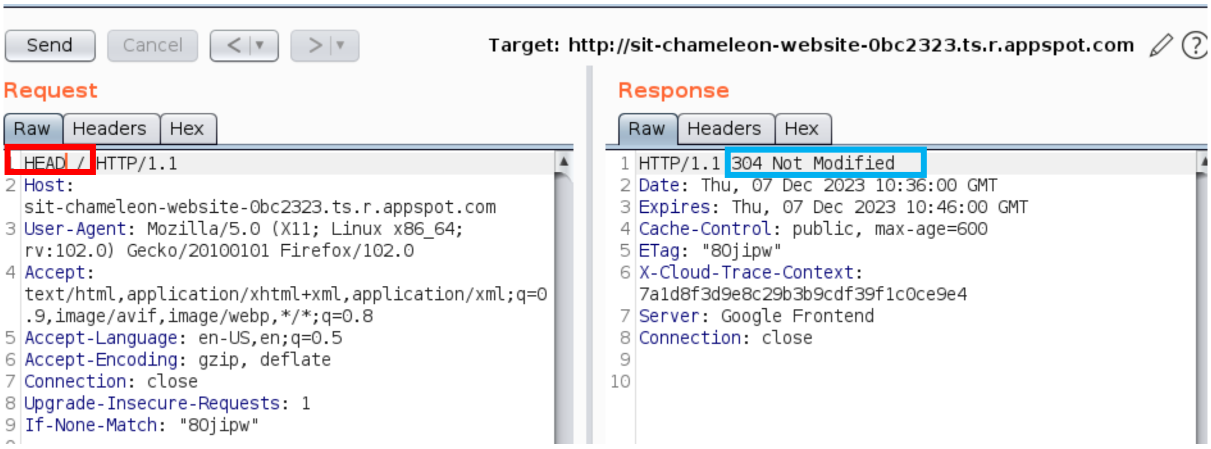


```
1  OPTIONS / HTTP/1.1
2  Host:
   sit-chameleon-website-0bc2323.ts.r.appspot.com
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64;
   rv:102.0) Gecko/20100101 Firefox/102.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0
   .9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Upgrade-Insecure-Requests: 1
9  If-None-Match: "80jipw"
10
11
```

```
1  HTTP/1.1 412 Precondition Failed
2  Date: Thu, 07 Dec 2023 10:26:18 GMT
3  Expires: Thu, 07 Dec 2023 10:36:18 GMT
4  Cache-Control: public, max-age=600
5  ETag: "80jipw"
6  X-Cloud-Trace-Context: c12db45d8d80869a800d7021019b564
7  Content-Type: text/html; charset=UTF-8
8  Server: Google Frontend
9  Content-Length: 296
10 Connection: close
11
12
13 <html>
     <head>
14     <meta http-equiv="content-type" content="text/html
15     <title>
         412 Precondition Failed
       </title>
16   </head>
17   <body text=#000000 bgcolor=#ffffff>
18     <h1>
         Error: Precondition Failed
       </h1>
19     <h2>
         Your client sent a conditional header that was n
       </h2>
20     <h2>
       </h2>
21   </body>
   </html>
22
```

Figure 3: Screenshot showing the OPTIONS HTTP Method in Burp Suite, and the Associated Response.

The next attack that was attempted, was the HEAD method. The HEAD method (shown in a red box on the request on the left, in Figure 4) will only display the header of the response, without the content. This has resulted in the header of the response being shown, as expected. The page would be displayed to the user successfully, as the response code is code 304 (shown in blue), which indicates that the page is the same as before it was reloaded, and thus, does not need to retrieve the webpage from the server again (Mozilla, 2023a). The HEAD method, is not an attack, but was tested as it is a common method. It shows the same information as a GET method, but without the content.



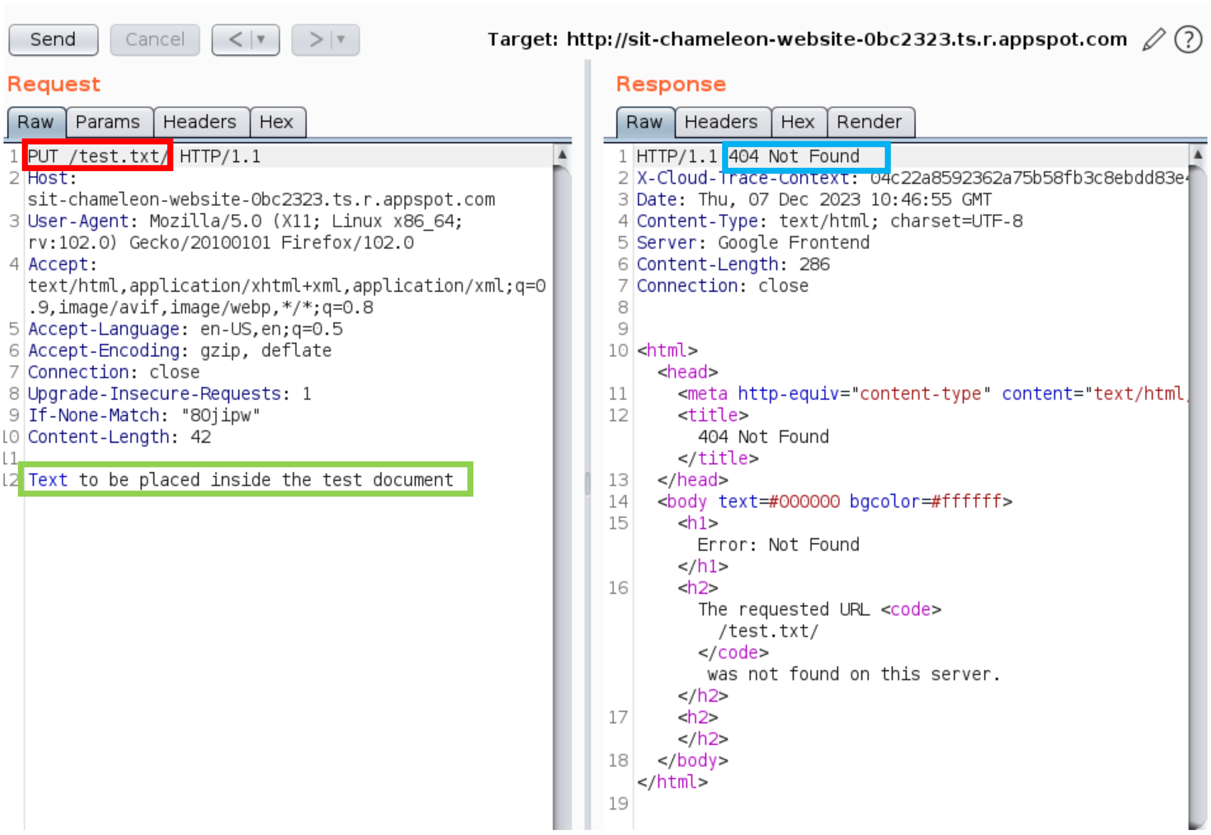
Target: http://sit-chameleon-website-0bc2323.ts.r.appspot.com

```
HEAD / HTTP/1.1
Host:
sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
```

```
HTTP/1.1 304 Not Modified
Date: Thu, 07 Dec 2023 10:36:00 GMT
Expires: Thu, 07 Dec 2023 10:46:00 GMT
Cache-Control: public, max-age=600
ETag: "80jipw"
X-Cloud-Trace-Context:
7a1d8f3d9e8c29b3b9cdf39f1c0ce9e4
Server: Google Frontend
Connection: close
```

Figure 4: Screenshot showing the HEAD HTTP Method in Burp Suite, and the Associated Response.

The next attack that will be tested, is the PUT method. The PUT method uploads a file to the webserver. Malicious files can be uploaded to the webserver, using this method. In Figure 5, it can be seen that the method was set to PUT (in red box). The */test.txt* is the file that will be created on the webserver. The content to be placed inside the *test.txt* document, is the content at the bottom of the request (the text: "*Text to be placed inside the test document"*) in green. The response that was provided for this request, consisted of the HTTP code 404 Not Found. This indicates that the *test.txt* file was not located on the webserver (Mozilla, 2023c). An accompanying error is displayed to the user. As a result of this, the HTTP method attack with this method was unsuccessful, meaning that the website is resistant to this attack.



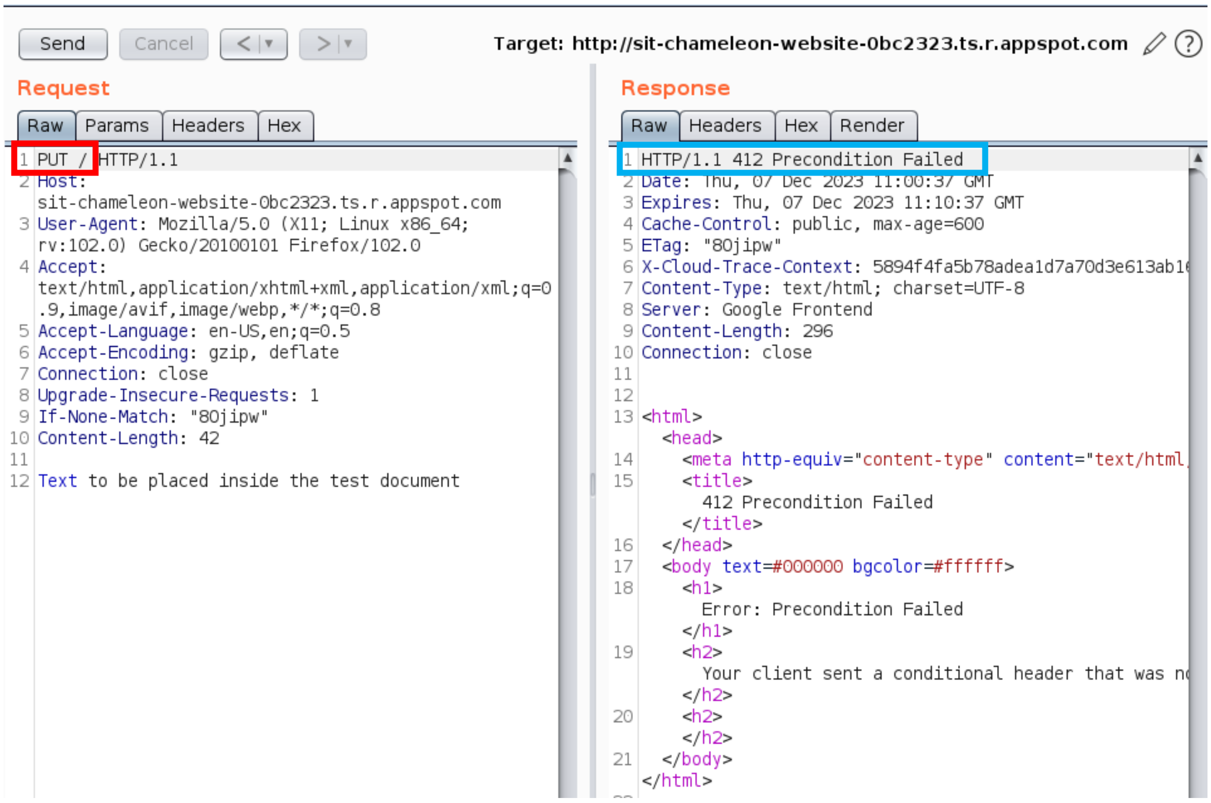
Target: http://sit-chameleon-website-0bc2323.ts.r.appspot.com

```
PUT /test.txt/ HTTP/1.1
Host:
sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
Content-Length: 42

Text to be placed inside the test document
```

```
HTTP/1.1 404 Not Found
X-Cloud-Trace-Context: 04c22a8592362a75b58fb3c8ebdd83e
Date: Thu, 07 Dec 2023 10:46:55 GMT
Content-Type: text/html; charset=UTF-8
Server: Google Frontend
Content-Length: 286
Connection: close


<html>
  <head>
    <meta http-equiv="content-type" content="text/html
    <title>
      404 Not Found
    </title>
  </head>
  <body text=#000000 bgcolor=#ffffff>
    <h1>
      Error: Not Found
    </h1>
    <h2>
      The requested URL <code>
        /test.txt/
      </code>
       was not found on this server.
    </h2>
    <h2>
    </h2>
  </body>
</html>
```

Figure 5: Screenshot showing the PUT HTTP Method in Burp Suite, and the Associated Response.

When the PUT method attack is repeated, but without a filename(as seen in Figure 6, in a red box), the response is the 412 response code (in blue), which indicates the webserver denied this request.

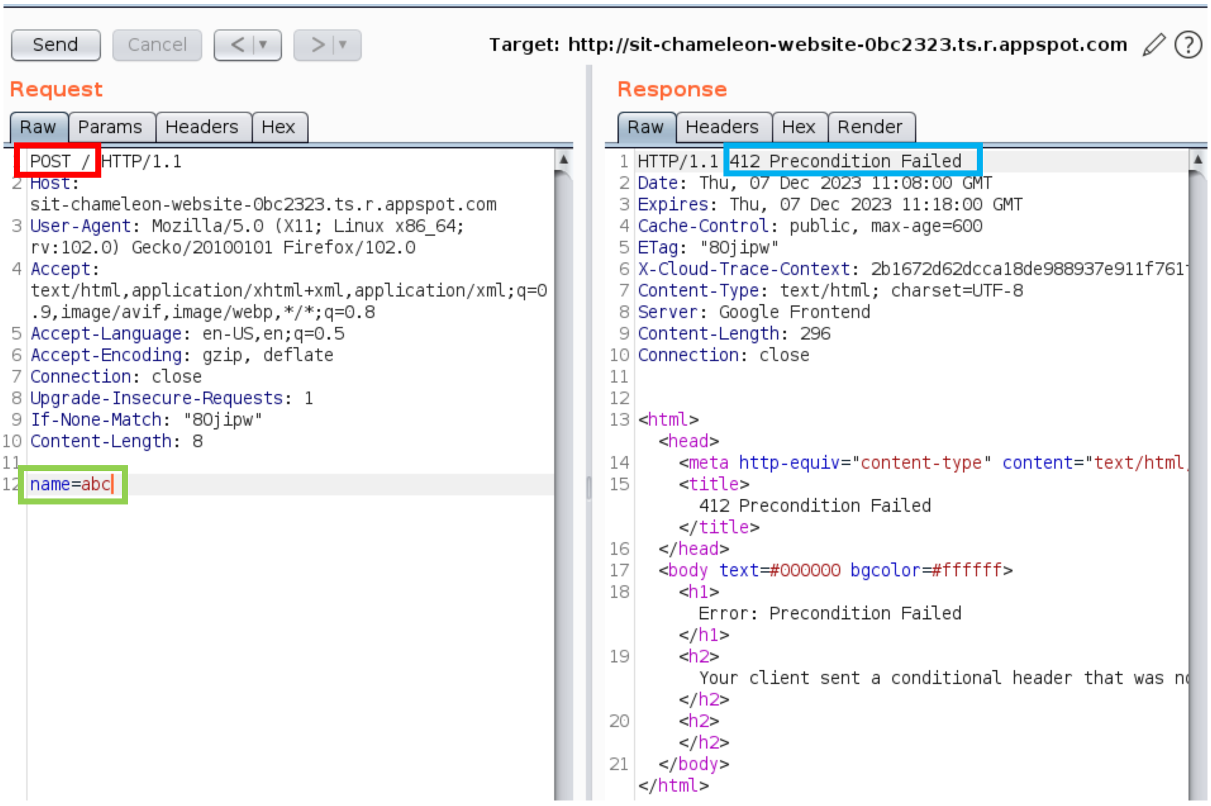


```
PUT / HTTP/1.1
Host:
sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
Content-Length: 42

Text to be placed inside the test document
```

```
HTTP/1.1 412 Precondition Failed
Date: Thu, 07 Dec 2023 11:00:37 GMT
Expires: Thu, 07 Dec 2023 11:10:37 GMT
Cache-Control: public, max-age=600
ETag: "80jipw"
X-Cloud-Trace-Context: 5894f4fa5b78adea1d7a70d3e613ab1
Content-Type: text/html; charset=UTF-8
Server: Google Frontend
Content-Length: 296
Connection: close


<html>
  <head>
    <meta http-equiv="content-type" content="text/html
    <title>
      412 Precondition Failed
    </title>
  </head>
  <body text=#000000 bgcolor=#ffffff>
    <h1>
      Error: Precondition Failed
    </h1>
    <h2>
      Your client sent a conditional header that was n
    </h2>
    <h2>
    </h2>
  </body>
</html>
```

Figure 6: Screenshot showing the PUT HTTP Method in Burp Suite, and the Associated Response.

The next method that was utilised, was the POST method, as shown in Figure 7. The GET method is used for retrieving pages, whereas the POST method is used to send data to the webserver. The POST method could be used for malicious activity, as it may allow malicious commands to be injected into the server. The POST method can be seen in the request in red, along with the content inside the body (*"name=abc"*), which is shown in green. The response code is 412, which indicates that the request was denied. This indicates that the website is hardened to prevent HTTP Method attacks.



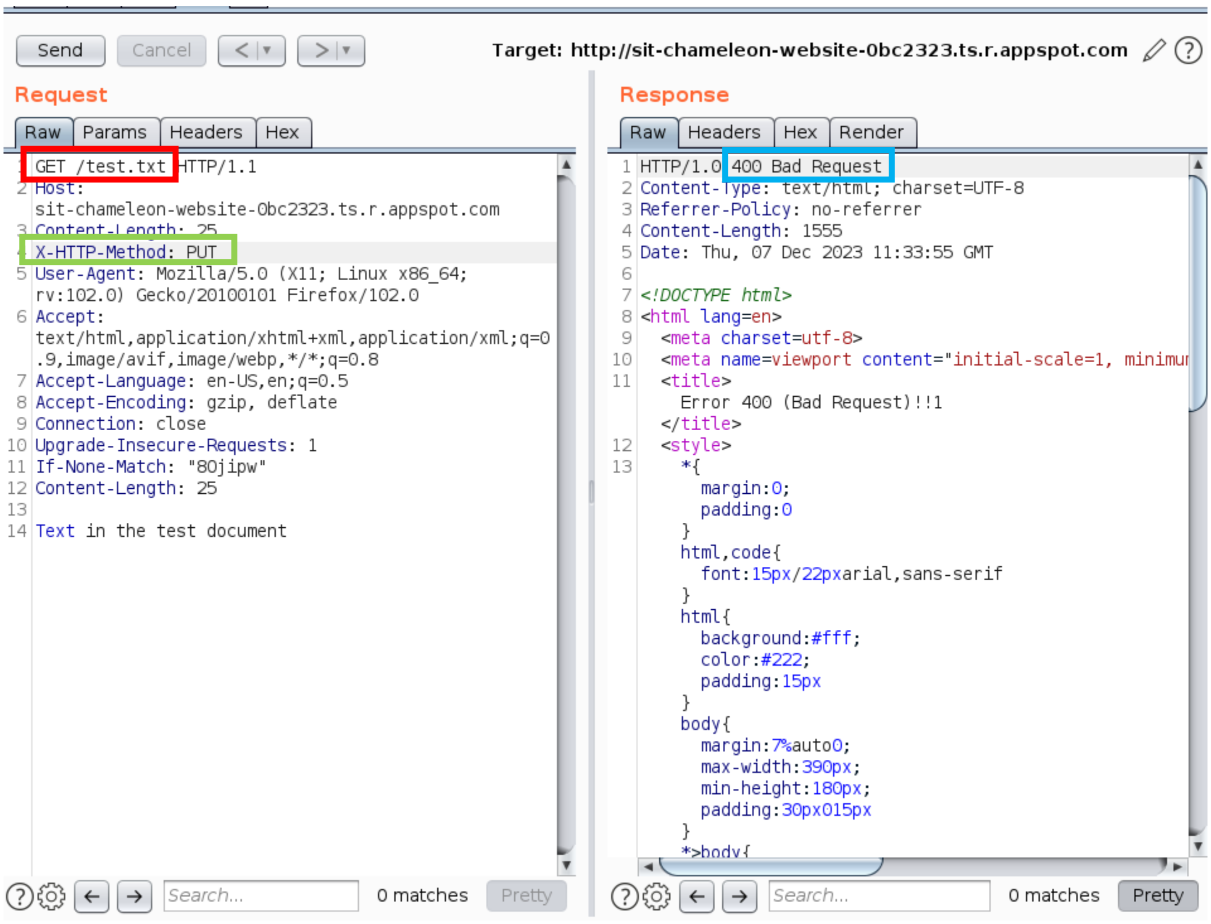
```
POST / HTTP/1.1
Host:
sit-chameleon-website-0bc2323.ts.r.appspot.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
Content-Length: 8

name=abc
```

```
HTTP/1.1 412 Precondition Failed
Date: Thu, 07 Dec 2023 11:08:00 GMT
Expires: Thu, 07 Dec 2023 11:18:00 GMT
Cache-Control: public, max-age=600
ETag: "80jipw"
X-Cloud-Trace-Context: 2b1672d62dcca18de988937e911f761
Content-Type: text/html; charset=UTF-8
Server: Google Frontend
Content-Length: 296
Connection: close


<html>
  <head>
    <meta http-equiv="content-type" content="text/html
    <title>
      412 Precondition Failed
    </title>
  </head>
  <body text=#000000 bgcolor=#ffffff>
    <h1>
      Error: Precondition Failed
    </h1>
    <h2>
      Your client sent a conditional header that was n
    </h2>
    <h2>
    </h2>
  </body>
</html>
```

Figure 7: Screenshot showing the POST HTTP Method in Burp Suite, and the Associated Response.

Another HTTP Method that can be used in a malicious manner is the DELETE method. This method will not be tested, to prevent accidental deletion of files on the webserver.

An approach to evade the countermeasures that are in place, is to add another parameter to the Header of the Request. This method involves using the method GET, and a file (shown in red in Figure 8), and then an additional parameter called *X-HTTP-Method* (shown in green), which will override the previous GET method. This resulted in the response displaying the code 400 Bad Request (shown in blue), which indicates that the request was malformed, or contained invalid syntax (Mozilla, 2023b).



Target: http://sit-chameleon-website-0bc2323.ts.r.appspot.com

Request

```
GET /test.txt HTTP/1.1
Host:
sit-chameleon-website-0bc2323.ts.r.appspot.com
Content-Length: 25
X-HTTP-Method: PUT
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: "80jipw"
Content-Length: 25

Text in the test document
```

Response

```
HTTP/1.0 400 Bad Request
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Content-Length: 1555
Date: Thu, 07 Dec 2023 11:33:55 GMT

<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimu
  <title>
    Error 400 (Bad Request)!!1
  </title>
  <style>
    *{
      margin:0;
      padding:0
    }
    html,code{
      font:15px/22pxarial,sans-serif
    }
    html{
      background:#fff;
      color:#222;
      padding:15px
    }
    body{
      margin:7%auto0;
      max-width:390px;
      min-height:180px;
      padding:30px015px
    }
    *>body{
```

Figure 8: Screenshot showing the GET HTTP Method, with an Overiding Method using PUT, in Burp Suite, and the Associated Response.

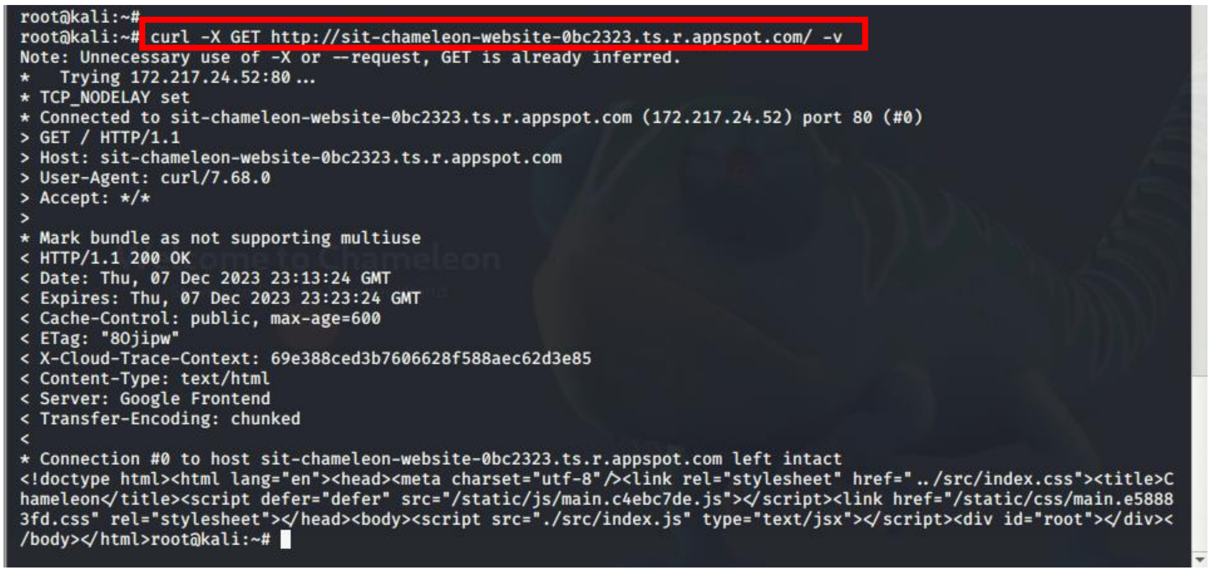
Curl was then used, to perform similar tests. Curl is a command line interface, rather than a graphical user interface, like Burp Suite. To understand how Curl works, a normal GET request was created as part of a command (shown in red in Figure 9), and sent to the webserver. It can be seen that the response header is displayed, and then the body contains the content of the webpage.

The components of the command are explained below:

-X: Specify the method to use.

GET: The method to use. Can be POST, PUT, DELETE.

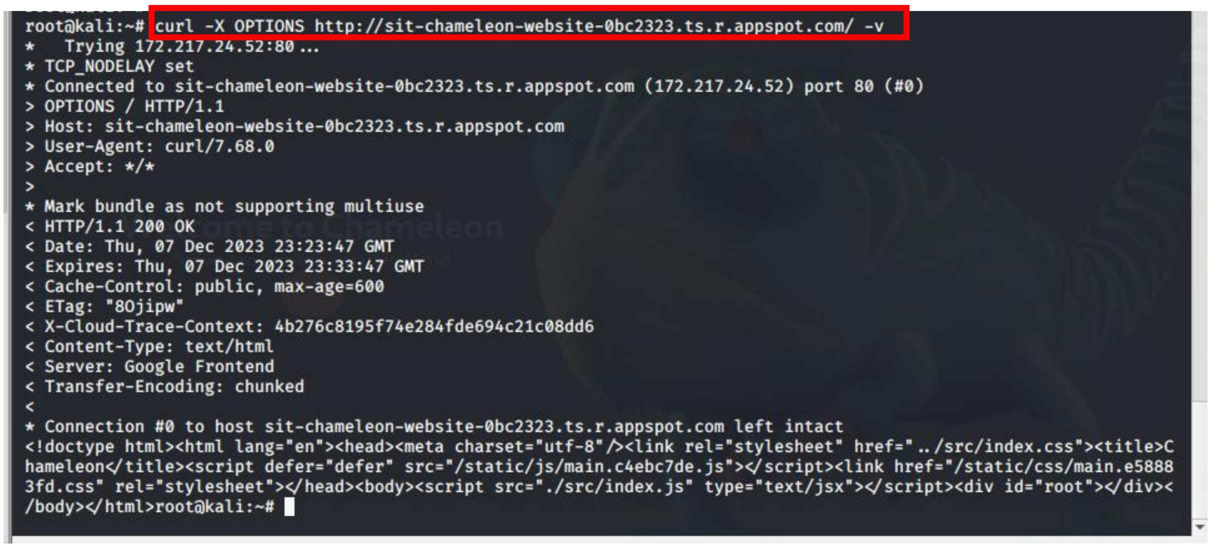
-v: Curl will produce more information in its responses, including display the header.



```
root@kali:~#
root@kali:~# curl -X GET http://sit-chameleon-website-0bc2323.ts.r.appspot.com/ -v
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 172.217.24.52:80...
* TCP_NODELAY set
* Connected to sit-chameleon-website-0bc2323.ts.r.appspot.com (172.217.24.52) port 80 (#0)
> GET / HTTP/1.1
> Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 07 Dec 2023 23:13:24 GMT
< Expires: Thu, 07 Dec 2023 23:23:24 GMT
< Cache-Control: public, max-age=600
< ETag: "8Ojipw"
< X-Cloud-Trace-Context: 69e388ced3b7606628f588aec62d3e85
< Content-Type: text/html
< Server: Google Frontend
< Transfer-Encoding: chunked
<
* Connection #0 to host sit-chameleon-website-0bc2323.ts.r.appspot.com left intact
<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="stylesheet" href="../src/index.css"><title>C
hameleon</title><script defer="defer" src="/static/js/main.c4ebc7de.js"></script><link href="/static/css/main.e5888
3fd.css" rel="stylesheet"></head><body><script src="./src/index.js" type="text/jsx"></script><div id="root"></div><
/body></html>root@kali:~#
```

Figure 9: Screenshot showing Curl Command using a GET Method.

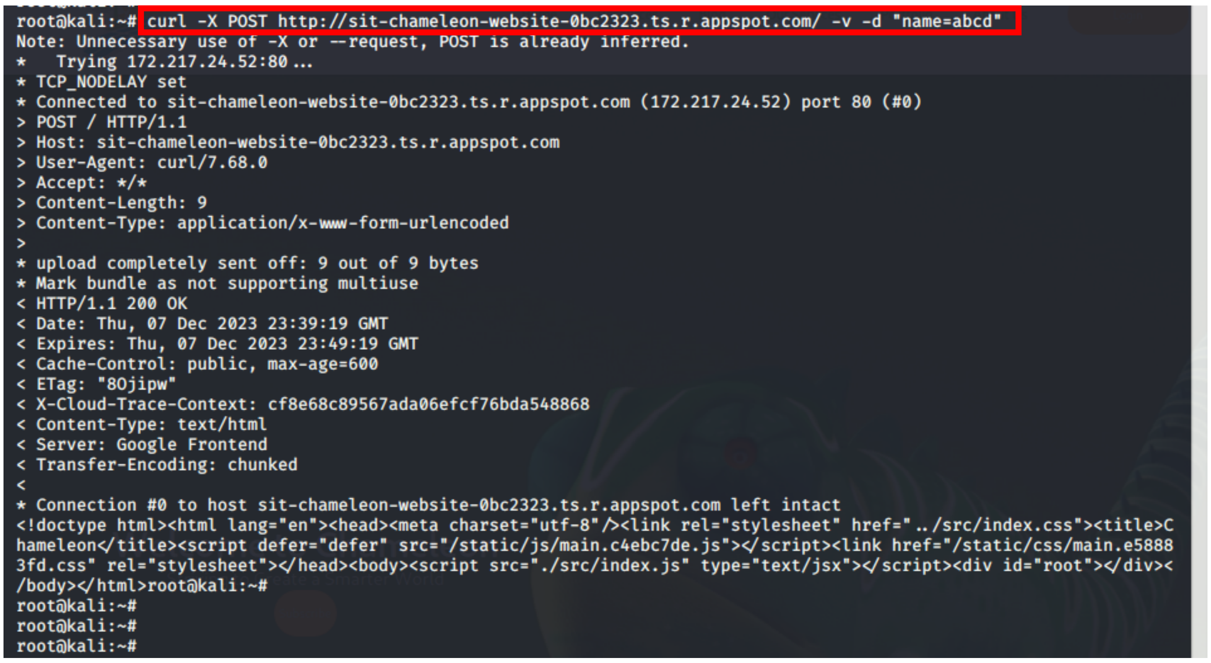
An OPTIONS request (see red box, in Figure 10) will be sent to the webserver through curl. The output was the same as using the GET method, and the response did not indicate the methods that it accepts. This indicates that the webserver has been hardened to protect against this attack. A similar command was tested, but with user data added to it (through adding "*-d "name=abcd""), but this returned the same result.*



```
root@kali:~# curl -X OPTIONS http://sit-chameleon-website-0bc2323.ts.r.appspot.com/ -v
*   Trying 172.217.24.52:80...
* TCP_NODELAY set
* Connected to sit-chameleon-website-0bc2323.ts.r.appspot.com (172.217.24.52) port 80 (#0)
> OPTIONS / HTTP/1.1
> Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 07 Dec 2023 23:23:47 GMT
< Expires: Thu, 07 Dec 2023 23:33:47 GMT
< Cache-Control: public, max-age=600
< ETag: "8Ojipw"
< X-Cloud-Trace-Context: 4b276c8195f74e284fde694c21c08dd6
< Content-Type: text/html
< Server: Google Frontend
< Transfer-Encoding: chunked
<
* Connection #0 to host sit-chameleon-website-0bc2323.ts.r.appspot.com left intact
<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="stylesheet" href="../src/index.css"><title>C
hameleon</title><script defer="defer" src="/static/js/main.c4ebc7de.js"></script><link href="/static/css/main.e5888
3fd.css" rel="stylesheet"></head><body><script src="./src/index.js" type="text/jsx"></script><div id="root"></div><
/body></html>root@kali:~#
```

Figure 10: Screenshot showing Curl Command using an OPTIONS Method.

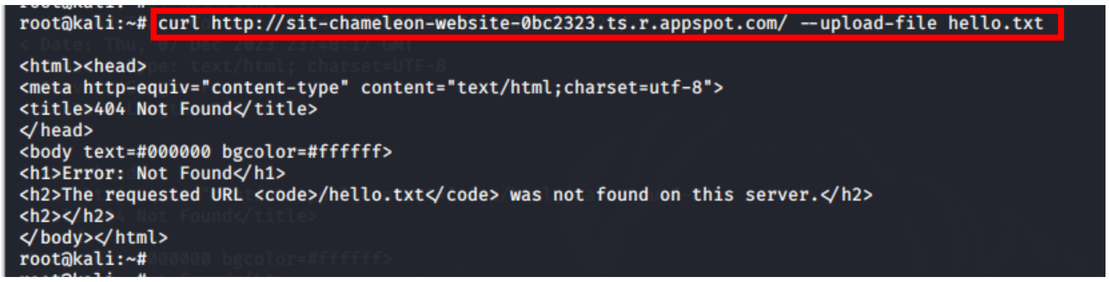
A POST request containing data was then sent to the webserver through Curl (See red box, in Figure 11). This request was successful, as the response code is 200, and the contents of the page was shown in the message body. Although this request was successful, it is believed that it behaved in a similar manner to a GET request, and loaded the webpage. I do not believe that it had a malicious impact on the website.



```
root@kali:~# curl -X POST http://sit-chameleon-website-0bc2323.ts.r.appspot.com/ -v -d "name=abcd"
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 172.217.24.52:80 ...
* TCP_NODELAY set
* Connected to sit-chameleon-website-0bc2323.ts.r.appspot.com (172.217.24.52) port 80 (#0)
> POST / HTTP/1.1
> Host: sit-chameleon-website-0bc2323.ts.r.appspot.com
> User-Agent: curl/7.68.0
> Accept: */*
> Content-Length: 9
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 9 out of 9 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 07 Dec 2023 23:39:19 GMT
< Expires: Thu, 07 Dec 2023 23:49:19 GMT
< Cache-Control: public, max-age=600
< ETag: "8Ojipw"
< X-Cloud-Trace-Context: cf8e68c89567ada06efcf76bda548868
< Content-Type: text/html
< Server: Google Frontend
< Transfer-Encoding: chunked
<
* Connection #0 to host sit-chameleon-website-0bc2323.ts.r.appspot.com left intact
<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="stylesheet" href="../src/index.css"><title>C
hameleon</title><script defer="defer" src="/static/js/main.c4ebc7de.js"></script><link href="/static/css/main.e5888
3fd.css" rel="stylesheet"></head><body><script src="./src/index.js" type="text/jsx"></script><div id="root"></div><
/body></html>root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
```

Figure 11: Screenshot showing Curl Command using a POST Method.

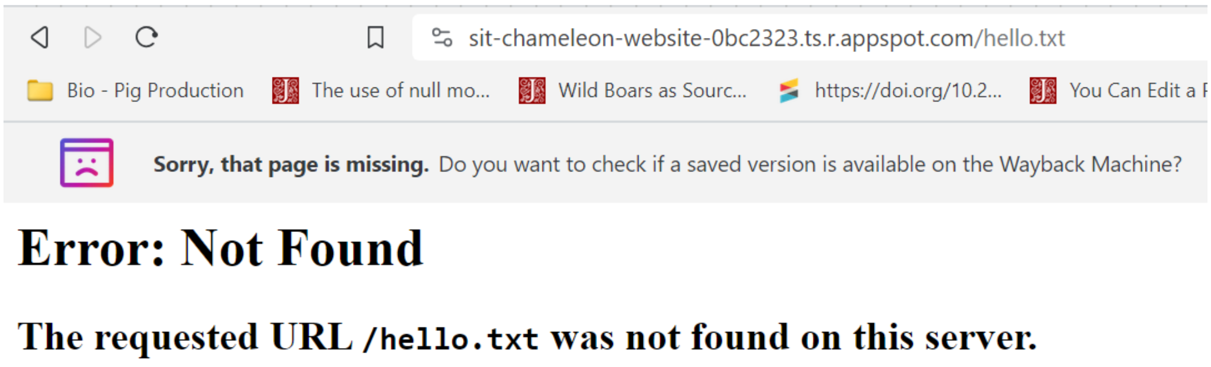
A PUT request was then tested in Curl, utilising the command "*--upload-file hello.txt*", shown in red, in figure 12. The purpose of this request, is to upload the file *hello.txt* to the webserver. This resulted in the response code 404 Not Found, which indicates that the webserver is unable to locate the file hello.txt. The body of the message also contains an error. Therefore, it was found that the Chameleon website is not vulnerable to PUT Method attacks.



```
root@kali:~# curl http://sit-chameleon-website-0bc2323.ts.r.appspot.com/ --upload-file hello.txt
<html><head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<title>404 Not Found</title>
</head>
<body text=#000000 bgcolor=#ffffff>
<h1>Error: Not Found</h1>
<h2>The requested URL <code>/hello.txt</code> was not found on this server.</h2>
<h2></h2>
</body></html>
root@kali:~#
```

Figure 12: Screenshot showing Curl Command using Attempting to use the PUT Method to upload a file to the server.

After attempting to upload the file, an attempt was made to view the file on the webpage, through entering the filename in the URL. This was tested on multiple pages of the website. Each attempt returned an error, stating that the file was not found on the server (See Figure 13). Therefore, it can be confirmed that the PUT Method to upload a file, was unsuccessful.



Sorry, that page is missing. Do you want to check if a saved version is available on the Wayback Machine?

Error: Not Found

The requested URL `/hello.txt` was not found on this server.

Figure 13: Screenshot showing the Attempt to View the Uploaded File.

Conclusion

As a result of the testing, it has been found that the Chameleon website is not vulnerable to HTTP Method attacks. A series of tests has been completed, utilising both Burp Suite, and Curl, and all tests were unsuccessful. A range of different methods were tested, including GET, POST, OPTIONS, HEAD, and PUT. The vulnerability that was tested, arises out of a misconfigured web server (OWASP Foundation, n.d.-b). Therefore, as the attack was unsuccessful, it can be concluded that the webserver was configured correctly for HTTP Methods. As the PUT and DELETE (DELETE was not tested, but the application is believed to be resistant to DELETE methods) methods were unsuccessful, an attack is not able to use this approach to upload files, or delete files on the web server.

As the testing found that the website is not vulnerable to this attack, no further actions need to be taken, in regards to this matter.

References

Mozilla (2023a) *304 Not Modified*, accessed 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/304

Mozilla (2023b) *400 Bad Request*, accessed 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400

Mozilla (2023c) *404 Not Found,* accessed 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404

Mozilla (2023d) *412 Precondition Failed*, accessed 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/412

OWASP Foundation (n.d.-a) *Test HTTP Methods*, accessed 2023. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods

OWASP Foundation (n.d.-b) *Test HTTP Methods*, accessed 2023. https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods