

Bypassing Basic Authentication

Exploiting HTTP Verb Tampering vulnerabilities is usually a relatively straightforward process. We just need to try alternate HTTP methods to see how they are handled by the web server and the web application. While many automated vulnerability scanning tools can consistently identify HTTP Verb Tampering vulnerabilities caused by insecure server configurations, they usually miss identifying HTTP Tampering vulnerabilities caused by insecure coding. This is because the first type can be easily identified once we bypass an authentication page, while the other needs active testing to see whether we can bypass the security filters in place.

The first type of HTTP Verb Tampering vulnerability is mainly caused by [Insecure Web Server Configurations](#), and exploiting this vulnerability can allow us to bypass the HTTP Basic Authentication prompt on certain pages.

Identify

When we start the exercise at the end of this section, we see that we have a basic [File Manager](#) web application, in which we can add new files by typing their names and hitting [enter](#):

However, suppose we try to delete all files by clicking on the red [Reset](#) button. In that case, we see that this functionality seems to be restricted for authenticated users only, as we get the following [HTTP Basic Auth](#) prompt:

As we do not have any credentials, we will get a [401 Unauthorized](#) page:

So, let's see whether we can bypass this with an HTTP Verb Tampering attack. To do so, we need to identify which pages are restricted by this authentication. If we examine the HTTP request after clicking the Reset button or look at the URL that the button navigates to after clicking it, we see that it is at [/admin/reset.php](#). So, either the [/admin](#) directory is restricted to authenticated users only, or only the [/admin/reset.php](#) page is. We can confirm this by visiting the [/admin](#) directory, and we do indeed get prompted to log in again. This means that the full [/admin](#) directory is restricted.

Exploit

To try and exploit the page, we need to identify the HTTP request method used by the web application. We can intercept the request in Burp Suite and examine it:

As the page uses a [GET](#) request, we can send a [POST](#) request and see whether the web page allows [POST](#) requests (i.e., whether the Authentication covers [POST](#) requests). To do so, we can right-click on the intercepted request in Burp and select [Change Request Method](#):

Once we do so, we can click [Forward](#) and examine the page in our browser. Unfortunately, we still get prompted to log in and will get a [401 Unauthorized](#) page if we don't provide the credentials:

So, it seems like the web server configurations do cover both [GET](#) and [POST](#) requests. However, as we have previously learned, we can utilize many other HTTP methods, most notably the [HEAD](#) method, which is identical to a [GET](#) request but does not return the body in the HTTP response. If this is successful, we may not receive any output, but the [reset](#) function should still get executed, which is our main target.

To see whether the server accepts [HEAD](#) requests, we can send an [OPTIONS](#) request to it and see what HTTP methods are accepted, as follows:

As we can see, the response shows [Allow: POST,OPTIONS,HEAD,GET](#), which means that the web server indeed accepts [HEAD](#) requests, which is the default configuration for many web servers. So, let's try to intercept the [reset](#) request again, and this time use a [HEAD](#) request to see how the web server handles it:

Once we change [POST](#) to [HEAD](#) and forward the request, we will see that we no longer get a login prompt or a [401](#) [Unauthorized](#) page and get an empty output instead, as expected with a [HEAD](#) request. If we go back to the [File Manager](#) web application, we will see that all files have indeed been deleted, meaning that we successfully triggered the [Reset](#) functionality without having admin access or any credentials:

Try testing other HTTP methods, and see which ones can successfully bypass the authentication prompt.

Cheat Sheet

Go to Questions

Table of Contents

Introduction to Web Attacks

HTTP Verb Tampering

Intro to HTTP Verb Tampering

Bypassing Basic Authentication

Bypassing Security Filters

Verb Tampering Prevention

Insecure Direct Object References (IDOR)

Intro to IDOR

Identifying IDORs

Mass IDOR Enumeration

Bypassing Encoded References

IDOR in Insecure APIs

Chaining IDOR Vulnerabilities

IDOR Prevention

XML External Entity (XXE) Injection

Intro to XXE

Local File Disclosure

Advanced File Disclosure

Blind Data Extraction

XXE Prevention

Skills Assessment

Web Attacks - Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Cheat Sheet

Go to Questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

+ 1 Try to use what you learned in this section to access the 'reset.php' page and delete all files. Once all files are deleted, you should get the flag.

HTB{4lw4t5_c0v3r_4ll_v3rb6}

Submit

Hint

Mark Complete & Next

Powered by HACKTHEBOX