Understanding the HTTP Protocol

posted in[**Penetration Testing**](https://www.hackingarticles.in/category/penetration-testing/), [**Website Hacking**](https://www.hackingarticles.in/category/website-hacking/) on [**July 1, 2017**](https://www.hackingarticles.in/understanding-http-protocol/) by [**Raj Chandel**](https://www.hackingarticles.in/author/raaz/)

SHARE

Save

HTTP (Hyper Text Transfer Protocol) is basically a client-server protocol, wherein the client (web browser) makes a request to the server and in return, the server responds to the request. The response by the server is mostly in the form of HTML formatted pages. HTTP protocol by default uses port 80, but the web server and the client can be configured to use a different port.

HTTP is a stateless protocol which means that the server does not retain the information from each user. HTTP is the backbone of the World Wide Web (www) and for it being stateless simply means that it does not remember each and every client that connects to the internet and it does not matter if a single user sends multiple requests one after another, they all will still be treated as independent requests by the server.

We are currently using HTTP 2, its predecessors were HTTP 1.0 and 1.1, and the major differences between 1.X and 2, at a higher level, are:

* Http 2 is binary and not textual
* Http 2 is multiplexed, it can use a single connection for parallelism, Http one, on the other hand, is based on ordering and blocking.
* Http 2 uses compression in its headers to reduce the overhead.
* Http2 gives servers the capability to “push” responses to client servers proactively.

HTTP works through different methods and these methods are:

**HTTP Request Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| GET | Used to retrieve information from the given URL |
| POST | Used to send data to the server, for example, customer information, file upload, etc. using HTML forms |
| DELETE | Delete a File of the specified URL |
| PUT | Uploads a File of the specified URL |
| TRACE | Trace on the jsp resource returns the content of the resource. |
| HEAD | GET only HTTP headers and no document body |
| OPTIONS | HTTP methods that the server supports |

There is a major difference between GET and POST method which people fail to understand. Once you understand these properly, you can manipulate and increase the security of your web application. The differences are as follows:

|  |  |
| --- | --- |
| **GET** | **POST** |
| Get request can be cached | Post request are never cached |
| Remain in the browser history | Do not remain in the browser history |
| It can be bookmarked | It cannot be bookmarked |
| Get request should never be used when dealing with sensitive data | The post should always be used for sensitive data |
| Get request has a length restriction | post request has no length restriction |
| Get request should be used to retrieve data |  |
| It is less secure | It is more secure |

An HTTP client sends an HTTP request to a server in the form of a request message which includes the following format



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | http://yahoo.com/  GET / HTTP/1.1  Host: yahoo.com  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:53.0) Gecko/20100101 Firefox/53.0  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  Accept-Language: en-US,en;q=0.5  Accept-Encoding: gzip, deflate  Connection: keep-alive  Upgrade-Insecure-Requests: 1 |

There are several fields in the header, but we will discuss the more important ones:

**Host**: This field is in the header and it is used to identify individual website by a hostname if they are sharing the same IP address. The client web browser also sets a user-agent string to identify the type and version of the browser.

**User-Agent**: This field is set correctly to its default values by the web browser, but it can be spoofed by the end user. This is usually done by the malicious user to retrieve contents designed for other types of web browsers.

**Cookie**: This field stores a temporary value shared between the client and server for session management.

**Referer**: This is another important field that you would often see when you are redirected from one URL to another. This field contains the address of the previous web page from which a link to the current page was followed. Attackers manipulate the Referer field using an XSS attack and redirect the user to a malicious website.

**Accept-Encoding**: This field defines the compression scheme supported by the client; gzip and Deflate are the most common ones. There are other parameters too, but they are of little use to penetration testers.

**Response**

**Response:** When a request is sent to the server; the server replies in the form of response. Following is an example of a response:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | HTTP/1.1 200 OK  Date: Sat, 10 Jun 2017 05:17:18 GMT  Set-Cookie: autorf=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/;  domain=in.yahoo.com  Content-Type: text/html; charset=UTF-8  Server: ATS  Expires: -1  Content-Length: 477864 |

**HTTP Response Code:** The Status-Code element is a 3-digit integer where the first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit

|  |  |  |
| --- | --- | --- |
| **Code** | **Meaning** | **Example** |
| 1xx | Information | *100*: server agrees to handle a client request. |
| 2xx | Success | *200*: request succeeded.  *204*: no client present. |
| 3xx | Redirection | *301*: page moved.  *304*: cached page still available. |
| 4xx | Client error | *403*: forbidden page.  *404*: page not found. |
| 5xx | Server error | *500*: internal server error.  *503*: try again later. |

**HTTP Version:** A server supporting HTTP version 1.1 will return the following version information

**Date:** The date and time that the message was originated

**Set-Cookie**: This field, if defined, will contain a random value that can be used by the server to identify the client and store temporary data

**Server**: This field is of interest to a penetration tester and will help in the recon phase of a test. It displays useful information about the web server hosting the website.   
**Content-Length**: This field will contain a value indicating the number of bytes in the body of the response. It is used so that the other party can know when the current request/response has finished.

**Source:**   
https://www.tutorialspoint.com/http/index.htm

https://en.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol  
https://www.httpwatch.com/httpgallery/

Multiple Ways To Exploiting HTTP Authentication

posted in[**Penetration Testing**](https://www.hackingarticles.in/category/penetration-testing/) on [**December 21, 2018**](https://www.hackingarticles.in/multiple-ways-to-exploiting-http-authentication/) by [**Raj Chandel**](https://www.hackingarticles.in/author/raaz/)

SHARE

Save

In this article, we will learn about how to configure the password-protected Apache Web Server to restrict from online visitors without validation so that we can hide some essential and critical information to the unauthenticated users and how to penetrate it’s the weak configuration to break its security and exploit it.

Table of Content

**Introduction to HTTP Basic Authentication**

* Lab Set\_up Requirement

**Setup Password Authentication**

* Installing the Apache utility Package
* Creating the Password File
* Configuring Access Control inside the Virtual Host Definition
* Configuring Access Control with .htaccess Files
* Confirm the Password Authentication

**Exploiting HTTP Authentication**

* xHydra
* Hydra
* Ncrack
* Medusa
* Metasploit
* Burp Suite

Introduction to HTTP Basic Authentication

In the context of an HTTP transaction, basic access authentication is a method for an HTTP user agent to provide a username and password when requesting.

HTTP Basic authentication (BA) implementation is the simplest technique for enforcing access controls to web resources because it doesn’t require cookies, session identifiers, or login pages; rather, HTTP Basic authentication uses standard fields in the HTTP header, obviating the need for handshakes.

The BA mechanism provides **no confidentiality protection** for the transmitted credentials. They are merely **encoded with Base64** in transit, but not encrypted or hashed in any way. HTTPS is, therefore, typically preferred used in conjunction with Basic Authentication.

For more details read from **wikipedia.org**

**Lab Setup Requirement**

Apache Server (Ubuntu 14.04)

Penetration Testing Machine (Kali Linux)

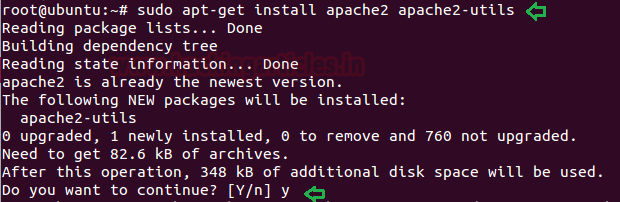
Set Up Password Authentication

**Installing the Apache utility Package**

Let’s start with the following command to install an Apache2 utility package called ‘htpasswd’. The htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users.



|  |  |
| --- | --- |
| 1 | sudo apt-get install apache2 apache2-utils |

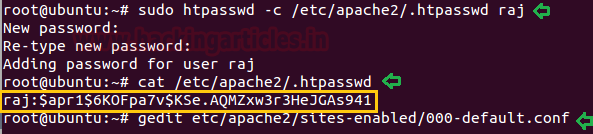


**Creating the Password File**

Now use the htpasswd command to create a password file that Apache will use to authenticate users and use a hidden file “.htpasswd” in our /etc/apache2 configuration directory to store password.



|  |  |
| --- | --- |
| 1  2 | sudo htpasswd -c /etc/apache2/.htpasswd raj  cat /etc/apache2/.htpasswd |



**Configuring Access Control inside the Virtual Host Definition**

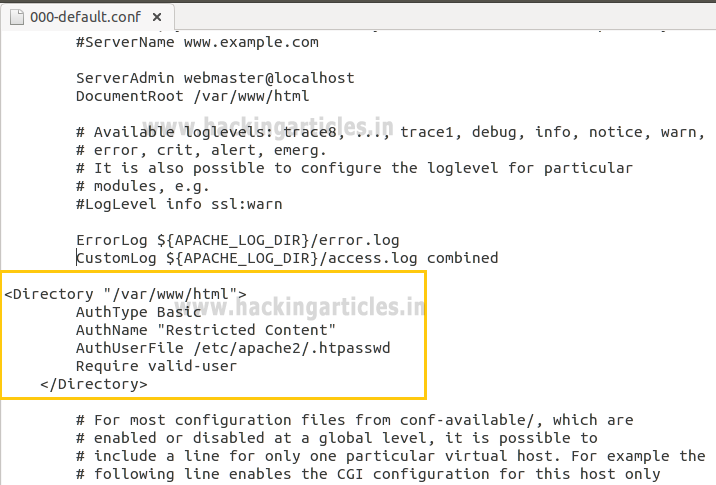
Now save the following configuration in 000-default.conf file.



|  |  |
| --- | --- |
| 1 | gedit etc/apache2/sites-enabled/000-default.conf |



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <Directory "/var/www/html">          AuthType Basic          AuthName "Restricted Content"          AuthUserFile /etc/apache2/.htpasswd          Require valid-user      </Directory> |

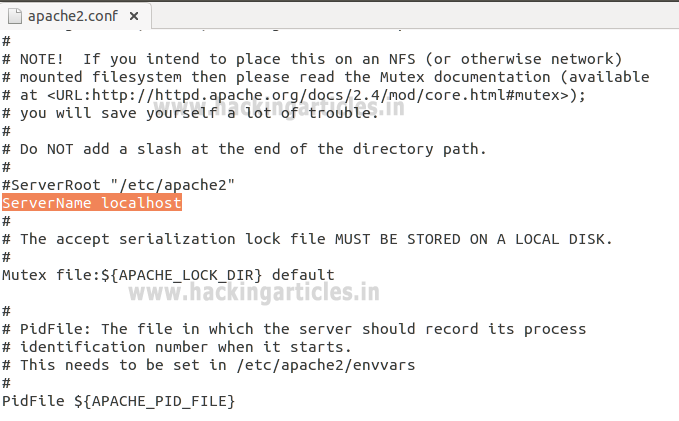


**Configuring Access Control with .htaccess Files**

Open the main Apache configuration file to enable password protection using .htaccess files and add the following line as highlighted.



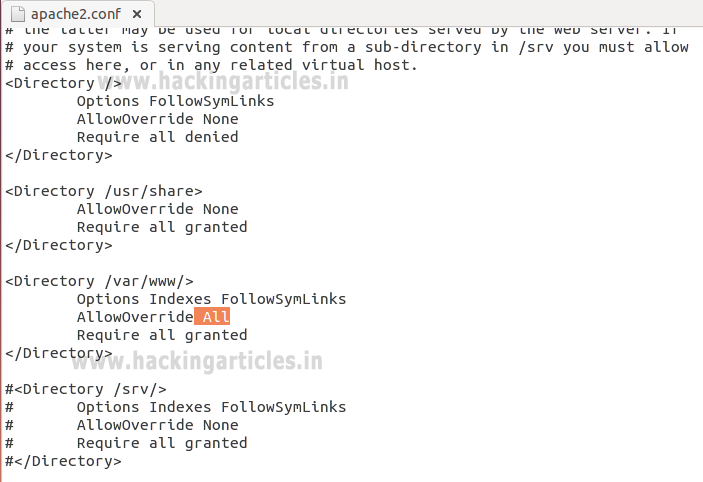
|  |  |
| --- | --- |
| 1  2 | sudo gedit /etc/apache2/apache2.conf  ServerName localhost |



Enable .htaccess processing by changing the AllowOverride directive “**None**” to “**All**” in the block for the /var/www directory and then save the file and restart the apache service.



|  |  |
| --- | --- |
| 1  2  3  4  5 | <Directory /var/www/>      Options Indexes FollowSymLinks      AllowOverride All      Require all granted  </Directory> |



Next, you need to add an htaccess file to the directory you wish to restrict. Here, I want to restrict the entire website which is could be through /var/www/html, but you can place this file in any directory **where you wish to restrict access:**



|  |  |
| --- | --- |
| 1  2  3  4 | AuthType Basic  AuthName "Restricted Content"  AuthUserFile /etc/apache2/.htpasswd  Require valid-user |



|  |  |
| --- | --- |
| 1 | sudo service apache2 restart |

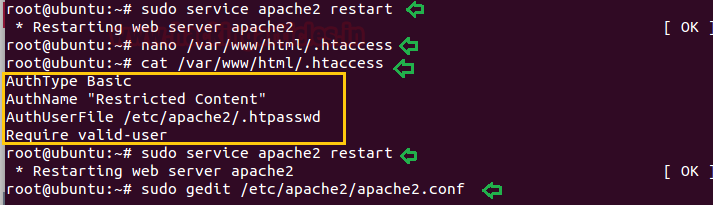
While configuring .htaccess file we had added few options for the block directory. Let’s see what this configuration denotes.

**AuthType Basic**: This will set up a basic authentication for our site.

**AuthName** “Restricted Contents”: This will show the name of the authentication at the prompt.

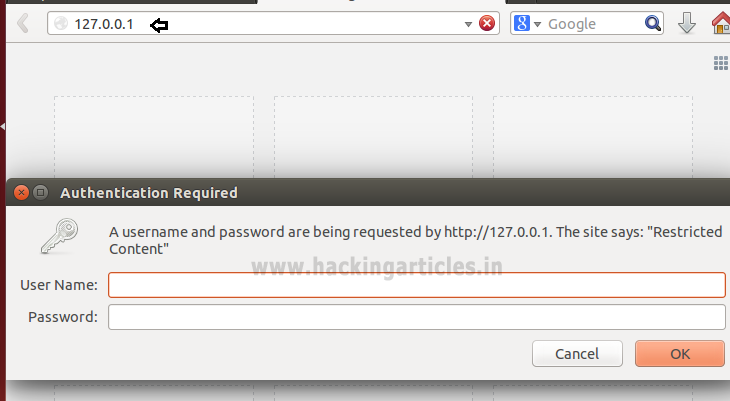
**AuthUserFile /etc/apache2/**.htpasswd : This will show the location of the authentication file.

**Require Valid-user:** This will be used by one user who has confirmed their authentication who are permitted to access the website.



Confirm the Password Authentication

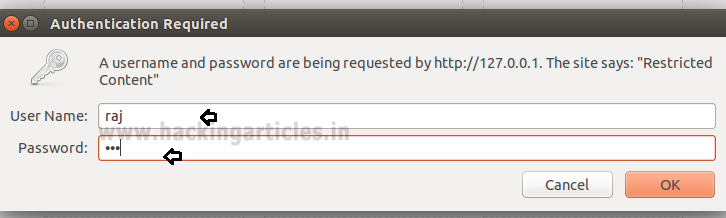
Try to access your restricted content in a web browser to confirm that your content is protected. I will be accessible with a username and password prompt that looks like this:



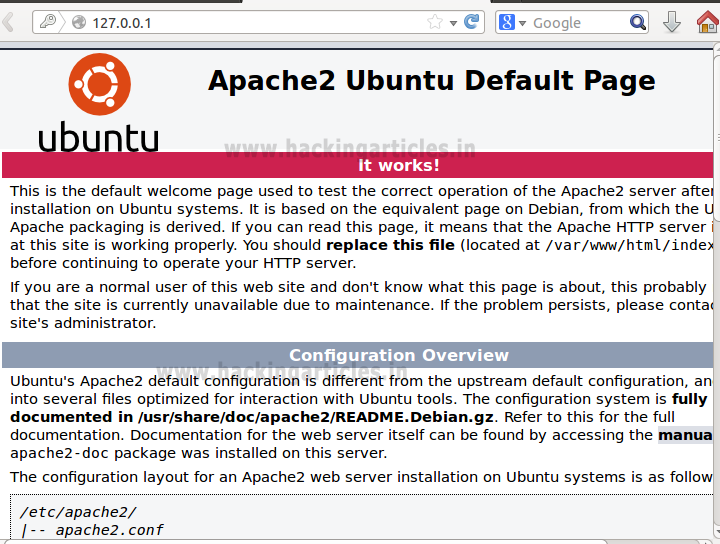
If you will try to access the website without authentication or canceled the Required Authentication page then it will displace 401 error Unauthorized Access.



If you are valid users and try to access password protected website by using the valid credential, for example, we had created an account with raj: 123 to access apache HTTP service.



As you can observe that, now we are able to access the content of the website.

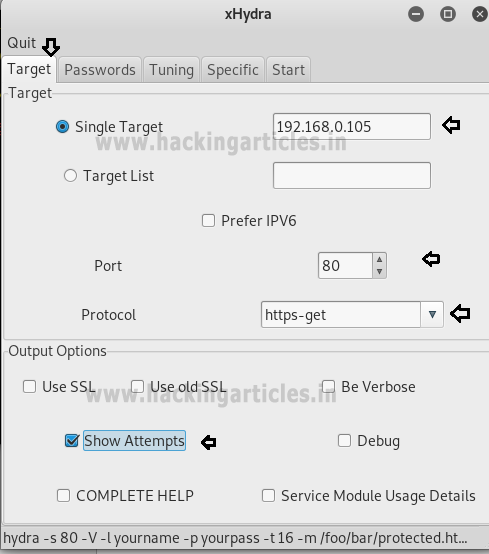


Exploiting HTTP Authentication

xHydra

This is the graphical version to apply dictionary attack via FTP port to hack a system. For this method to work:

Open **xHydra** in your Kali. And select **Single Target option** and there give the IP of your victim PC. And select **HTTP** in the box against **Protocol option** and give the port number **80** against the **port option**.



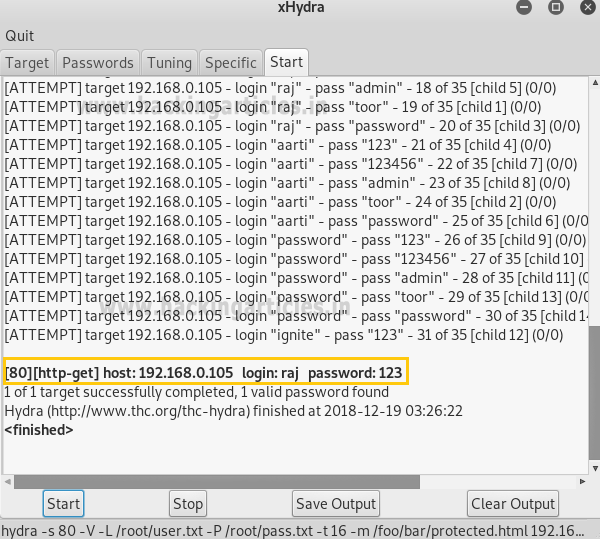
Now, go to **Passwords tab** and select **Username List** and give the path of your text file, which contains usernames, in the box adjacent to it.

Then select Password List and give the path of your text file, which contains all the passwords, in the box adjacent to it.



After doing this, go to the Start tab and click on the **Start** button on the left.

Now, the process of dictionary attack will start. Thus, you will obtain the username and password of your victim.



**Hydra**

Hydra is often the tool of choice. It can perform rapid dictionary attacks against more than 50 protocols, including telnet, FTP, HTTP, https, SMB, several databases, and much more

Now, we need to choose wordlist. As with any dictionary attack, the wordlist is key. Kali has numerous wordlists built right in.

Run the following command

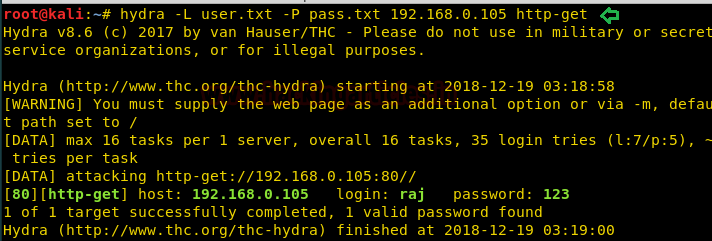


|  |  |
| --- | --- |
| 1 | hydra -L user.txt -P pass.txt 192.168.0.105 http-get |

**-L:** denotes the path for username list

**-P:**  denotes the path for the password list

Once the commands are executed it will start applying the dictionary attack and so you will have the right username and password in no time. As you can observe that we had successfully grabbed the HTTP username as raj and password as 123.



Ncrack

Ncrack is a high-speed network authentication cracking tool. It was built to help companies secure their networks by proactively testing all their hosts and networking devices for poor passwords.

Run the following command



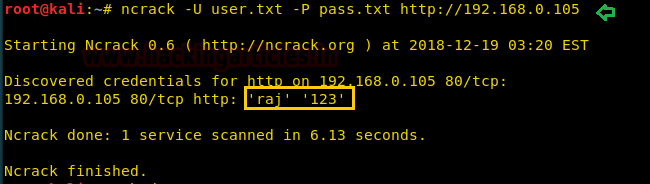
|  |  |
| --- | --- |
| 1 | ncrack -U user.txt -P pass.txt http://192.168.0.105 |

Here

**-U:** denotes the path for username list

**-P:**  denotes the path for the password list

As you can observe that we had successfully grabbed the HTTP username as raj and password as 123.



Medusa

Medusa is intended to be a speedy, massively parallel, modular, login brute-forcer. It supports many protocols: AFP, CVS, FTP, HTTP, IMAP, rlogin, SSH, Subversion, and VNC to name a few

Run the following command



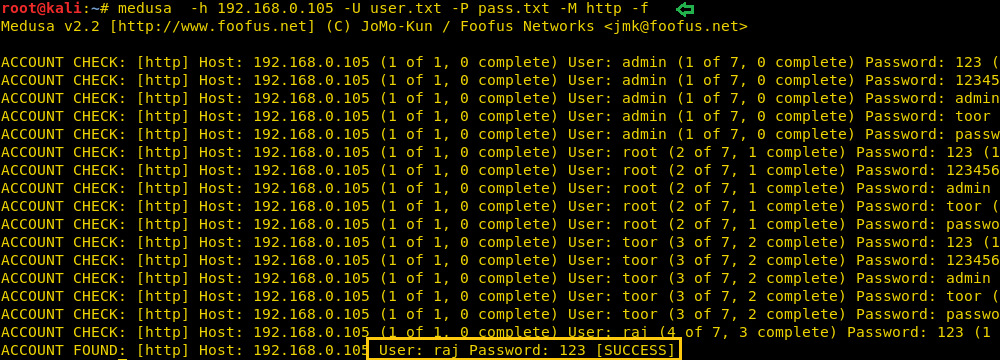
|  |  |
| --- | --- |
| 1 | medusa -h 192.168.0.105 -U user.txt -P pass.txt -M http -f |

Here

**-U:** denotes the path for username list

**-P:**  denotes the path for the password list

As you can observe that we had successfully grabbed the HTTP username as raj and password as 123.

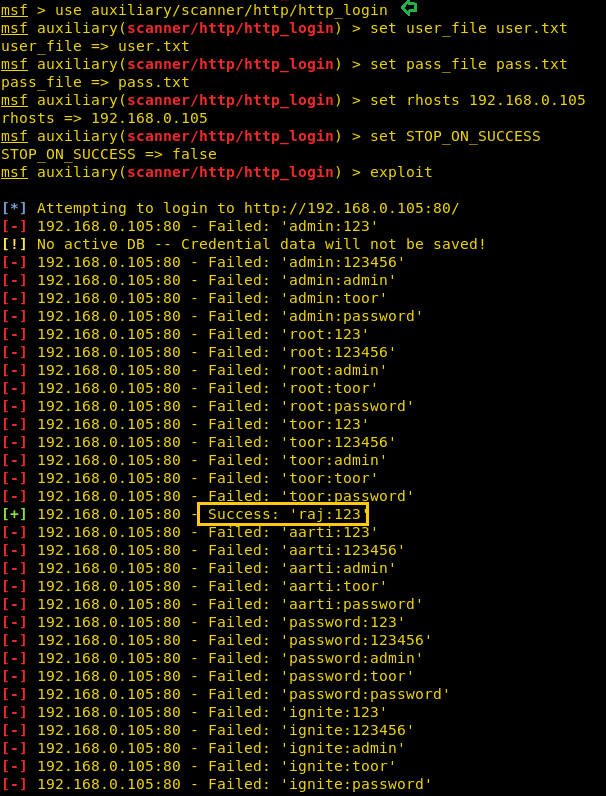


Metasploit

This module attempts to authenticate to an HTTP service. Open Kali terminal type**msfconsole** and then type:

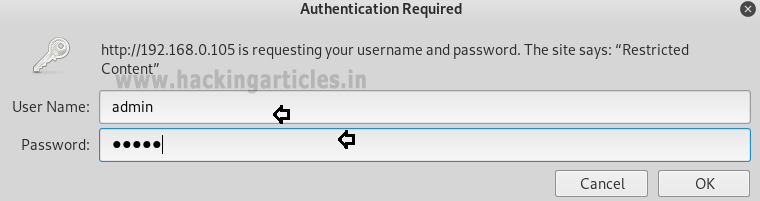


|  |  |
| --- | --- |
| 1  2  3  4  5  6 | use auxiliary/scanner/http/http\_login  msf auxiliary(scanner/http/http\_login) > set user\_file user.txt  msf auxiliary(scanner/http/http\_login) > set pass\_file pass.txt  msf auxiliary(scanner/http/http\_login) > set rhosts 192.168.0.105  msf auxiliary(scanner/http/http\_login) >  set stop\_on\_success true  msf auxiliary(scanner/http/http\_login) > exploit |



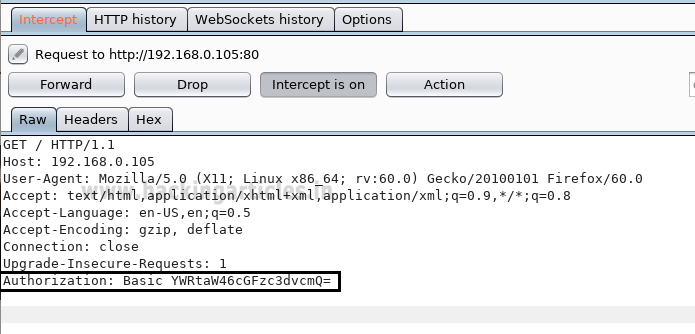
Burp Suite

Now here I had just typed the random value for authentication in order to fetch the request through Burp Suite. So before you sent the request to server **turn on** the **burp suite** and **select proxy tab** then, **click** on the **intercept is on** after then send the user authentication by clicking ok

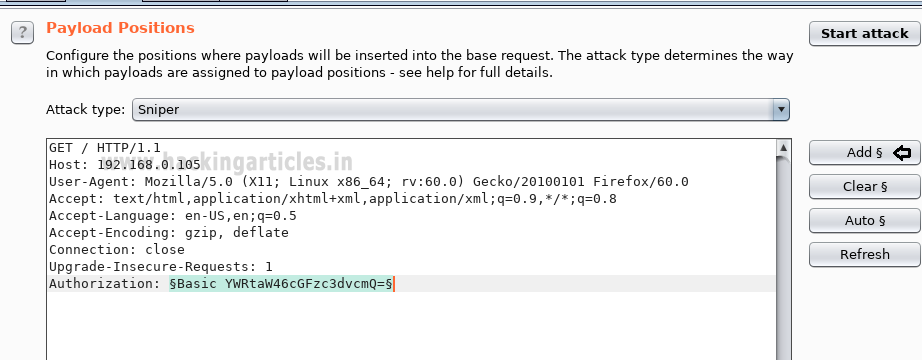


Thus the sent request is captured by burp suite which you can see in the given below image. In the screenshot, I had highlighted some value in the last line. Here it says the type of authentication provided is basic and if you have read above theory of basic authentication I had described that it is encoded in base64.

Now time to generate the encoded value for authentication inside the burp suite. **Click** on **action** tab **select send to intruder** for HTTP Fuzzing attack.

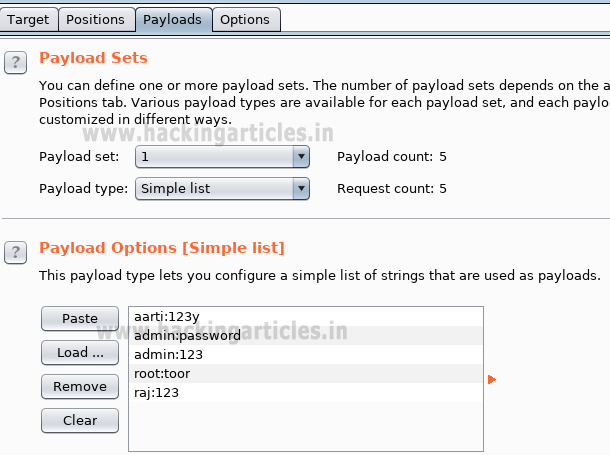


Now open intruder frame and click on the position. Configure the position where payload will be inserted into the request. The attack type determines the way in which the payload is assigned to the payload position. Now **select** “the encoded value of authentication” for payload position and click to **ADD** button on the left side of the frame.



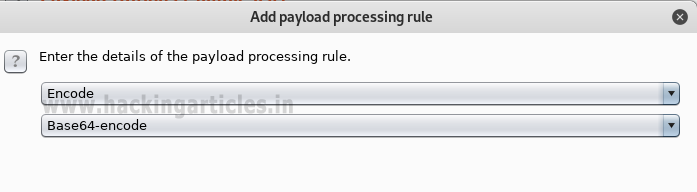
The base64 encoded value of Authentication is a combination of username and password now the scenario is to generate the same encoded value of authentication with the help of user password dictionary, Therefore, I have made a dictionary which contains both user password names in a text file.

In order to use the dictionary as payload click on **payload tab** under intruder**;** now **load** your dictionary which contains user password names from **payload options**.



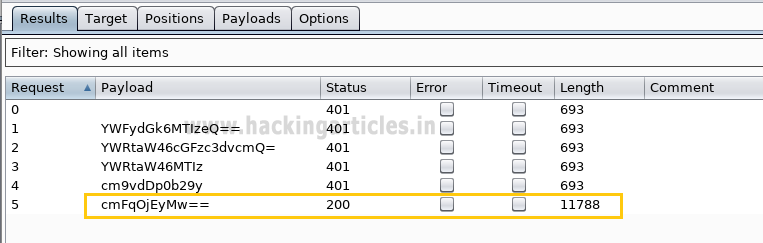
But we want to send a request in the encoded value of our payload. To encode your payload click on **ADD button** available under payload processing.

 A new dialog box will generate to select the rule to choose an encode option from the list; now select **base64** from drag down the list of URL encode key character for payload processing.

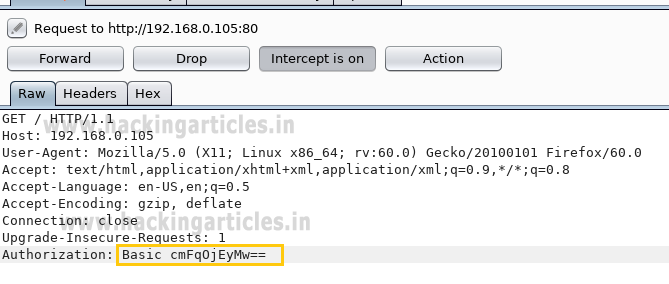


This will start a brute force attack and try to match string for user authentication. In the screenshot, you can observe the status “200 OK” and length “11788” of the highlighted value is different from the rest of the values. This means we can use this encoded value to bypass the user authentication, which occurs from request number 5. Now check the username and password on the 5th line in the dictionary. In the dictionary I found raj: 123 have matching authentication.

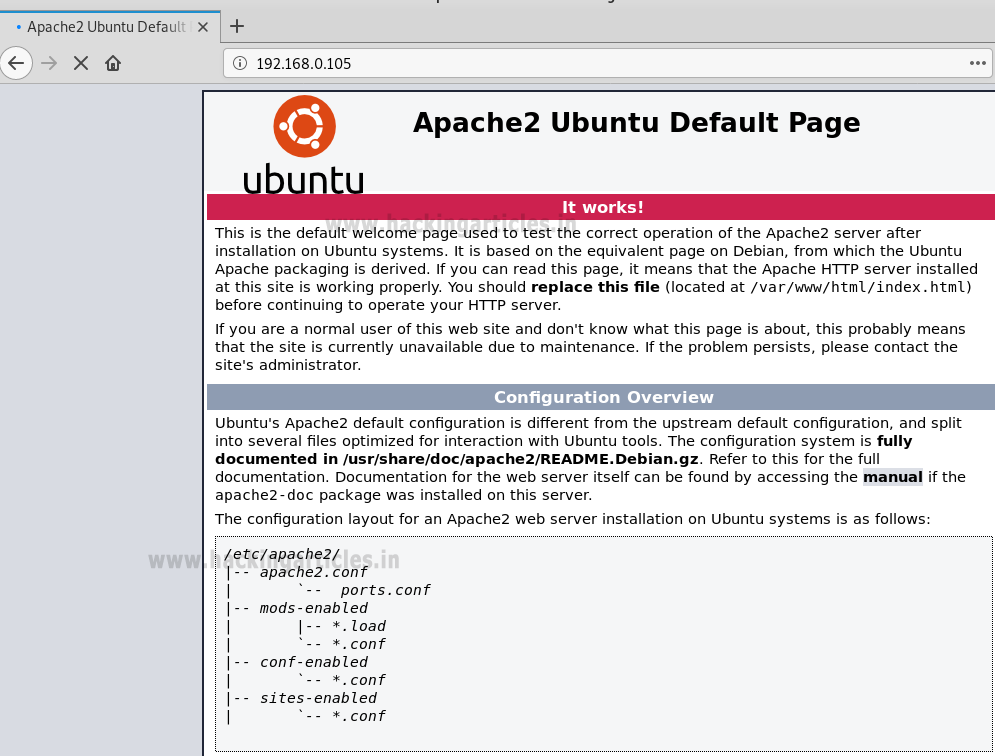
Or you can also use this encoded Auth value to bypass the Apache HTTP authentication page via burp suite intercepted data.



Copy the above auth value and paste replace it with intercepted authorization a shown in below and forward the request to access restricted content.



Booom!!! Here we have successfully access the content of the website.



Hope you people have enjoyed this article and learned how weak configuration security can easily breach and the unauthorized person can access the restrict content of your website.

**Author:**AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact[**here**](https://www.linkedin.com/in/aarti-singh-353698114)

Multiple Ways to Detect HTTP Options

posted in[**Kali Linux**](https://www.hackingarticles.in/category/kali-linux/), [**Penetration Testing**](https://www.hackingarticles.in/category/penetration-testing/) on [**October 5, 2018**](https://www.hackingarticles.in/multiple-ways-to-detect-http-options/) by [**Raj Chandel**](https://www.hackingarticles.in/author/raaz/)

SHARE

Save

Hi Friends, today we will walk through various HTTP Protocol methods and the tools used to extract those available HTTP methods in a web server. As we are already aware that the HTTP protocol comprises of a number of methods that can be utilized to not only gather the information from the web server but can also perform specific actions on the web server. These techniques and methods are helpful for the web application developers in the deployment and testing stage of web applications.

GET and POST is the most well-known methods that are used to access and submit information provided by a web server, respectively. HTTP Protocol allows various other methods as well, like PUT, CONNECT, TRACE, HEAD, DELETE. These methods can be used for malicious purposes if the web server is left misconfigured and hence poses a major security risk for the web application, as this could allow an attacker to modify the files stored on the web server.

**OPTIONS:** The OPTIONS method is used to request the available HTTP methods on a web server.

GET: GET request is the most common and widely used methods for the websites. This method is used to retrieve the data from the web server for a specific resource. As the GET method only requests for the data and doesn’t modify the content of any resources, it’s considered to be safe.

POST: POST requests are used to send (or submit) the data to the web server so as to create or update a resource. The information sent is stored in the request body of the HTTP request and processed further. An example illustrating the same is “Contact us” form page on a website. When we fill a form and submit it, the input data is then stored in the response body of the request and sent across to the server.

PUT: The PUT method allows the end-user (client) to upload new files on the web server. An attacker can exploit it by uploading malicious files or by using the victim’s server as a file repository.

CONNECT: The CONNECT method could allow a client to use the web server as a proxy.

**TRACE:** This method echoes back to the client, the same string which has been sent across to the server, and is used mainly for debugging purposes.

**HEAD:** The HEAD method is almost similar to GET, however without the message-body in the response. In other words, if the HTTP request **GET /products** return a list of products, then the **HEAD /products** will trigger a similar HTTP request, however, won’t retrieve the list of products.

**DELETE:** This method enables 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 perform a DoS attack.

Now let us use some tools to identify the HTTP methods enabled or supported by the web server

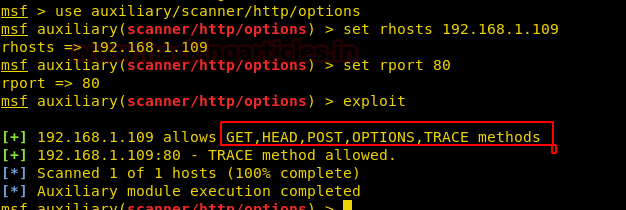
**Metasploit**

Metasploit Framework is a well-known platform for developing, testing, and executing exploits. It is an open source tool for performing various exploits against the target machines.

Metasploit has in-built auxiliary modules dedicated to scanning HTTP methods. Through the Metasploit framework command line (CLI), we can identify the HTTP Options available on the target URL as follows:



|  |  |
| --- | --- |
| 1  2  3  4 | use auxiliary/scanner/http/options  set rhosts 192.168.1.109  set rport 80  exploit |



**cURL**

cURL is a command line tool to get or send the data using the URL syntax and is compatible with various well-known protocols (HTTPS, FTP, SCP, LDAP, Telnet etc.) along with command line (CLI) options for performing various tasks (Eg: User authentication , FTP uploading , SSL connections etc). The cURL utility by default comes installed in most of the distributions. However if in case, cURL is not installed, then we can install the same via **apt-get install curl** command. For more details refer the below URL

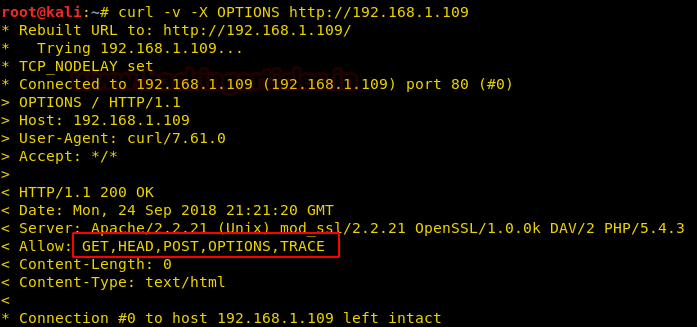
<https://www.hackingarticles.in/web-application-penetration-testing-curl/>

Through the cURL command we can identify the HTTP Options available on the target URL as follows :



|  |  |
| --- | --- |
| 1 | curl -v -X OPTIONS http://192.168.1.109 |

The screenshot displays the various types of allowed HTTP methods (GET, HEAD, POST, OPTIONS, TRACE), apart from other server-specific information (Server response, version details etc)



**Nikto**

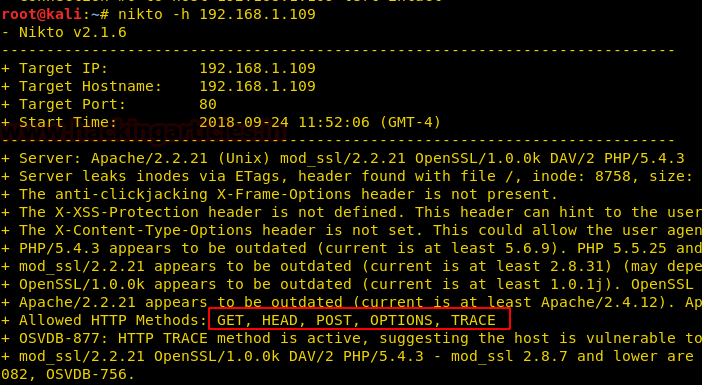
Nikto is a Web server scanner that tests Web servers for dangerous files/CGIs, outdated server software and other issues. It performs generic and server types of specific checks.

Through the Nikto command we can identify the HTTP Options available on the target URL as follows :



|  |  |
| --- | --- |
| 1 | nikto -h 192.168.1.109 |

The screenshot displays the various types of allowed HTTP methods (GET, HEAD, POST, OPTIONS, TRACE), apart from another detailed server specific information (Server response, version details etc)



**Nmap**

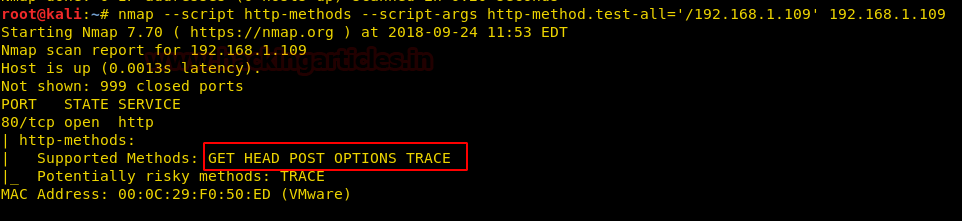
Nmap is a free and open-source security scanner, used to discover hosts and services on the network. This is another method of checking which HTTP methods are enabled by using an NMAP script called http-methods.nse, which can be obtained from <https://nmap.org/nsedoc/scripts/http-methods.html> .

Let us use NMAP command to enumerate all of the HTTP methods supported by a web server on the target URL as follows :



|  |  |
| --- | --- |
| 1 | nmap --script http-methods --script-args http-method.test-all ='/192.168.1.109' 192.168.1.109 |

The screenshot displays the various types of allowed HTTP methods (GET, HEAD, POST, OPTIONS, TRACE) along with highlighting the potential risk methods (i.e TRACE) out of them.



**Netcat**

Netcat is a utility tool having the capability to write and read data across TCP and UDP network connections, along with features like in-built port scanning, network debugging and file transfer etc.

Through the Netcat command we can identify the HTTP Options available on the target URL as follows :



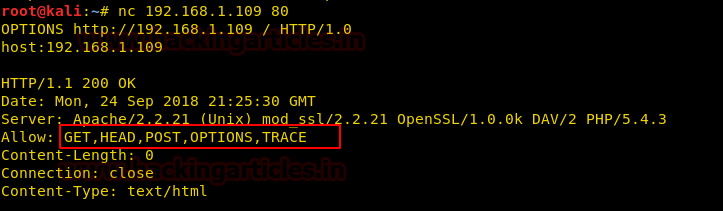
|  |  |
| --- | --- |
| 1 | nc 192.168.1.109 80 |

Press enter and the following options appear in the command line. Enter the server details as follows (and as highlighted in red )



|  |  |
| --- | --- |
| 1  2 | OPTIONS  http://192.168.1.109  / HTTP/1.0  host:192.168.1.109 |

The screenshot displays the various types of allowed HTTP methods (GET, HEAD, POST, OPTIONS, TRACE), apart from other server-specific information (Server response, version details etc)

****

**Burpsuite**

Burp Suite is a platform for performing various security testing for web applications, from initial mapping and analysis to identifying and exploiting application vulnerabilities.

As we are aware that the HTTP OPTIONS method provides us with the most effective way to discover the different methods allowed on an HTTP server. So, let us capture the URL request in Burpsuite GUI and change the HTTP method type in the Request section to OPTIONS, as seen below.

As shown, the RESPONSE from the web server not only displays the list of HTTP methods allowed, however also highlights the server version details (Eg: Apache/2.2.21 (Unix) mod\_ssl/2.2.21 OpenSSL 1.0.0/k DAV/2 PHP/5.4.3)