

Preignition Write-up

Prepared by: One-nine9

Introduction

In most environments, web servers play a big part in the infrastructure and in the daily processes of many departments. Web servers can sometimes be used strictly internally by employees, but most of the time can be found to be public-facing, meaning anyone from the Internet can access them to retrieve information and files from their hosted web pages. For the most part, the web pages hosted on the web servers are managed through their administrative panels, locked behind a log-in page. Let us think of an example:

You have decided to start your own blog and use WordPress to achieve this. If you are unfamiliar with WordPress, you can [read more about it here](#), but it is essentially a popular web application that allows you to easily manage the content you want to post for the rest of the world to read. Once installed, your WordPress website will have a public-facing side and a private-facing one, the latter being your administrative panel hosted at `www.yourwebsite.com/wp-admin`. This page is locked behind a log-in screen.

A quick note: you can learn more about hacking WordPress websites by checking out the [Hacking WordPress](#) module on HTB Academy!

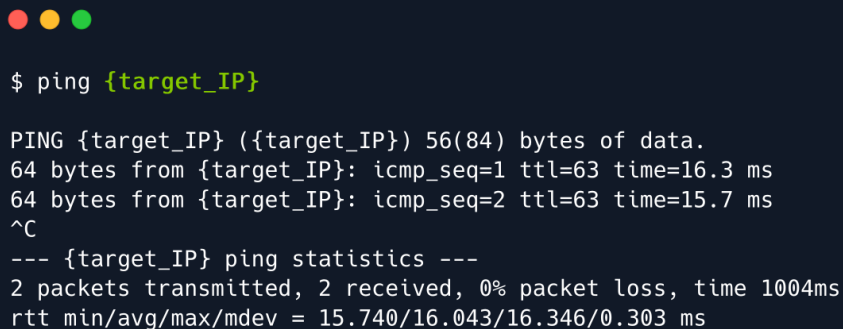


Once you, as an administrator of the WordPress site, log into its' admin panel, you will have access to a myriad of controls, ranging from content uploading mechanisms, to theme selection, custom script editing for specific pages, and more. The more you learn about WordPress, the more you will see how this is a vital part of a successful pentest, as some of these mechanisms could be outdated and come with critical flaws that would allow an attacker to gain a foothold and subsequently pivot through the network with ease.

Thus, we conclude that Web enumeration, specifically directory busting (dir busting), is one of the most essential skills any Penetration Tester must possess. While manually navigating websites and clicking all the available links may reveal some data, most of the links and pages may not be published to the public and, hence, are less secure. Suppose we did not know the `wp-admin` page is the administrative section of the WordPress site we exemplified above. How else would we have found it out if not for web enumeration and directory busting?

Enumeration

As previously, we start by verifying connectivity to the target. We can take the IP address of the target machine from the Starting Point lab page and paste it into the terminal and fire-up the ping command as a first step. After two successful replies, we can interrupt the ping command, as we are satisfied with the connection quality. We do not always need to run commands for a long time. Sometimes, getting a snippet of the result or an overview instead of a detailed report is more beneficial to our time efficiency than the alternative.



```
$ ping {target_IP}

PING {target_IP} ({target_IP}) 56(84) bytes of data.
64 bytes from {target_IP}: icmp_seq=1 ttl=63 time=16.3 ms
64 bytes from {target_IP}: icmp_seq=2 ttl=63 time=15.7 ms
^C
--- {target_IP} ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 15.740/16.043/16.346/0.303 ms
```

Immediately after, we can follow up with a preliminary scan of the target. Using nmap and the appropriate service version detection switch, we scan the IP address for any open ports and services.

```
-sV : Probe open ports to determine service/version info
```

```
$ sudo nmap -sV {target_IP}
```

```
Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-25 01:39 BST
```

```
Nmap scan report for {target_IP}
```

```
Host is up (0.053s latency).
```

```
Not shown: 999 closed tcp ports (reset)
```

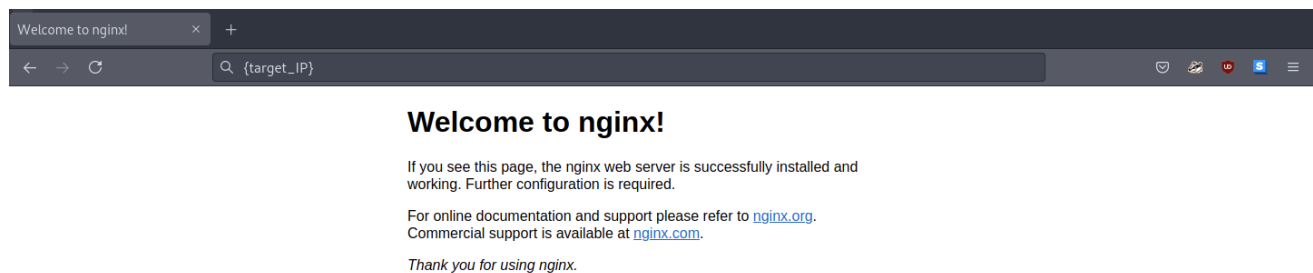
```
PORT      STATE SERVICE VERSION
```

```
80/tcp    open  http    nginx 1.14.2
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
```

```
Nmap done: 1 IP address (1 host up) scanned in 12.34 seconds
```

From the scan result, a single entry is shown and catches our attention. It is an http service running on port 80, signaling that this target might be hosting some explorable web content. To look at the contents ourselves, we can open a web browser of our choice and navigate to the target's IP address in the URL bar at the top of the window. This will automatically address the target's port 80 for the client-server communication and load the web page's contents.



At the top of the page, we observe the mention of the nginx service. After researching basic information about nginx and its purpose, we conclude that our target is a web server. Web servers are hosts on the target network which have the sole purpose of serving web content internal or external users, such as web pages, images, videos, audio files, and other types. Typically, a web server is accessible from the Internet to allow for the stored content to be explored by the online public for many reasons: shopping, providing and

requesting services, banking, reading the news, and more.

What we are looking at on our browser screen is the default post-installation page for the nginx service, meaning that there is the possibility that this web application might not be adequately configured yet, or that default credentials are used to facilitate faster configuration up to the point of live deployment. This, however, also means that there are no buttons or links on the web page to assist us with navigation between web directories or other content.

When browsing a regular web page, we use these elements to move around on the website. However, these elements are only links to other directories containing other web pages, which get loaded in our browser as if we manually navigated to them using the URL search bar at the top of the browser screen. Knowing this, could we attempt to find any "hidden" content hosted on this webserver?

The short answer is yes, but to avoid guessing URLs manually through the browser's search bar, we can find a better solution. This method is called dir busting, short for directory busting. For this purpose, we will be using the tool called `gobuster`, which is written in Go. If you do not have gobuster installed on your machine yet, you can follow the instructions below to install it successfully. Pwnbox already comes pre-equipped with gobuster and all the necessary tools to finish any lab on Hack The Box.

Installing gobuster

First, you need to make sure you have Go installed on your Linux distribution, which is the programming language used to write the gobuster tool. Once all the dependencies are satisfied for Go, you can proceed to download and install gobuster. In order to install Go, you need to input the following command in your terminal window:

```
sudo apt install golang-go
```

Once that installation is complete, you can proceed with installing gobuster. If you have a [Go](#) environment ready to go (at least go 1.16), it is as easy as typing in the following command in your terminal:

```
go install github.com/OJ/gobuster/v3@latest
```

In case this fails, you can always compile the tool from its' source code by running the following commands:

```
sudo git clone https://github.com/OJ/gobuster.git
cd gobuster
go get && go build
go install
```

If these installation instructions are unclear, please follow the gobuster creator's readme file on everything related to the tool by [following this link](#).

Using gobuster

In order to start our dir busting, we will need to discover what capabilities gobuster has and which ones can assist us. By looking at the tool's help page, by typing in the `gobuster -h` command in our terminal, we receive a list of all possible switches for the tool and their description.

Usage:

```
gobuster dir [flags]
```

Flags:

| | |
|--|--|
| <code>-f, --add-slash</code> | Append / to each request |
| <code>-c, --cookies string</code> | Cookies to use for the requests |
| <code>-e, --expanded</code> | Expanded mode, print full URLs |
| <code>-x, --extensions string</code> | File extension(s) to search for |
| <code>-r, --follow-redirect</code> | Follow redirects |
| <code>-H, --headers stringArray</code> | Specify HTTP headers, <code>-H 'Header1: val1' -H 'Header2: val2'</code> |
| <code>-h, --help</code> | help for dir |
| ..[Output omitted].. | |
| <code>-u, --url string</code> | The target URL |
| <code>-a, --useragent string</code> | Set the User-Agent string (default "gobuster/3.1.0") |
| <code>-U, --username string</code> | Username for Basic Auth |
| <code>-d, --discover-backup</code> | Upon finding a file search for backup files |
| <code>--wildcard</code> | Force continued operation when wildcard found |

Global Flags:

| | |
|----------------------------------|---|
| <code>-z, --no-progress</code> | Don't display progress |
| <code>-o, --output string</code> | Output file to write results to (defaults to stdout) |
| <code>-q, --quiet</code> | Don't print the banner and other noise |
| <code>-t, --threads int</code> | Number of concurrent threads (default 10) |
| <code>--delay duration</code> | Time each thread waits between requests (e.g. 1500ms) |

```
-v, --verbose          Verbose output (errors)
-w, --wordlist string   Path to the wordlist
```

In our case, we will only need to use the following:

```
dir : specify we are using the directory busting mode of the tool
-w : specify a wordlist, a collection of common directory names that are typically used
for sites
-u : specify the target's IP address
```

```
$ sudo gobuster dir -w /usr/share/wordlists/dirb/common.txt -u {target_IP}
```

```
[sudo] password for {username}:
```

```
Gobuster v3.1.0
```

```
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
```

```
[+] Url:          http://{target_IP}
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:      /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:    gobuster/3.1.0
[+] Timeout:      10s
```

```
2021/09/14 14:30:31 Starting gobuster in directory enumeration mode
```

In the wordlist we used, the entry `admin.php` was present, and from our results below, we can see that there is indeed a page with the same name present on the target. What gobuster did here was make several connection attempts to the target using the HTTP GET method (requesting web pages from the webserver) with different URL variations, such as:

```
GET http://{target_IP}/kittens
GET http://{target_IP}/hackers
GET http://{target_IP}/admin.php
GET http://{target_IP}/hackthebox
GET http://{target_IP}/yourespecial
```

Out of all the variations found in the wordlist we specified, `admin.php` existed and was returned to us in the output as seen below, signaling that the webpage exists and we can navigate to it manually to check out its contents:

```
$ sudo gobuster dir -w /usr/share/wordlists/dirb/common.txt -u {target_IP}
```

```
[sudo] password for {username}:
```

```
Gobuster v3.1.0
```

```
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
```

```
[+] Url: http://{target_IP}
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s
```

```
2021/09/14 14:30:31 Starting gobuster in directory enumeration mode
```

```
/admin.php (Status: 200) [Size: 999]
```

```
2021/09/14 14:30:35 Finished
```

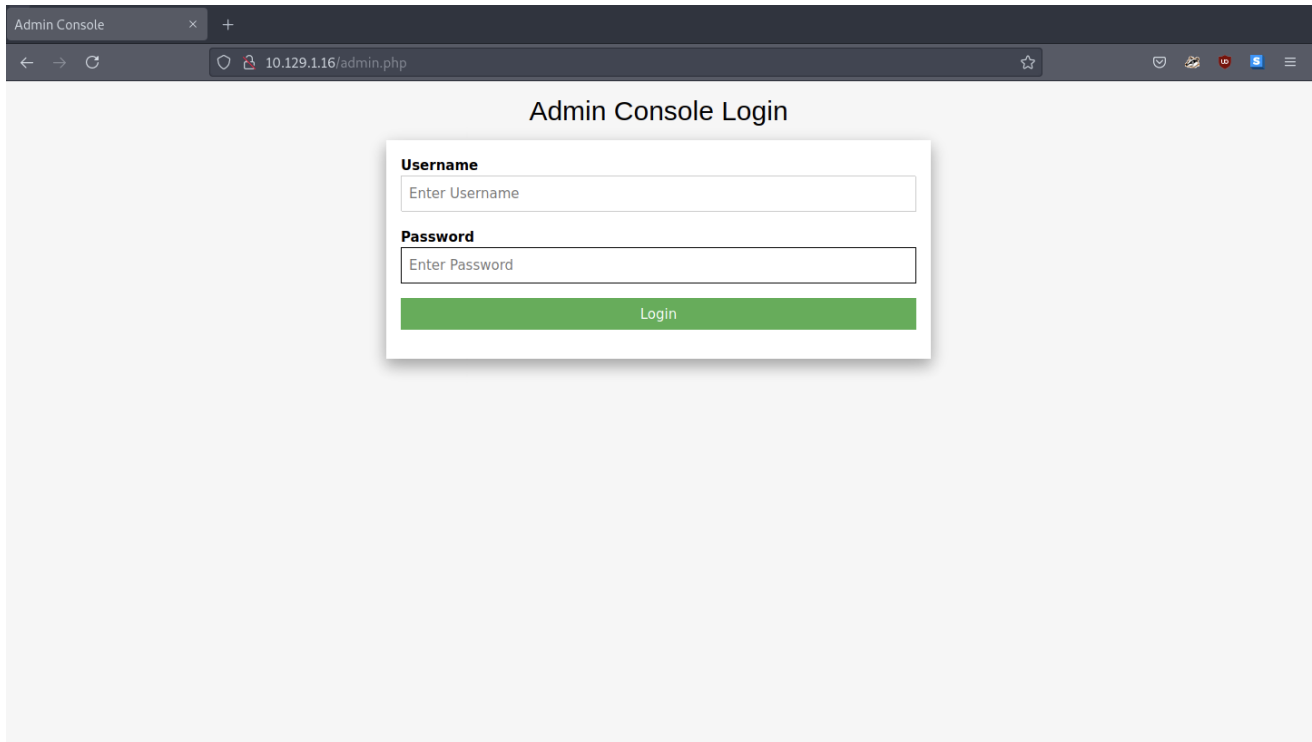
Now that we have received the result we needed from gobuster, we can put it away for now. If you want to delve even deeper into directory busting and web enumeration, you can follow up with our module on HTB Academy with Ffuf! Ffuf is just another tool like gobuster, around which we have developed a comprehensive module on HTB Academy. If you want to learn more about web directories and other web services, you can read up on our [Attacking Web Applications with Ffuf](#) module on HTB Academy.

Attacking Web Applications with Ffuf



Foothold

Navigating to the newly found link manually through our web browser can be done by inputting the following address in our URL search bar. Once we proceed with this, we are met with an administrative panel for the website. It asks us for a username and password to get past the security check, which could prove problematic in normal circumstances.



Usually, in situations such as this one, we would need to fire up some brute-forcing tools to attempt logging in with multiple credentials sets for an extended period of time until we hit a valid log-in since we do not have any underlying context about usernames and passwords that might have been registered on this web site as valid administrative accounts. But first, we can try our luck with some default credentials since this is a fresh nginx installation. We are betting that it might have been left unconfigured at the time of our assessment. Let us try logging in with the following credentials:

```
admin admin
```

Admin Console Login

Username

admin

Password

•••••

Login

We seem to be successful! The log-in worked, and we were presented with our flag. This concludes Tier 0, the most elementary of targets. Do not forget to take a look at the suggested modules in HTB Academy to get a better understanding of the tactics and vectors used so far in the assessments, and follow up with Tier I, which will offer less detailed explanations and more technicality in the assessment process!

Admin Console Login

Congratulations! Your flag is: 6483bee07c1c1d57f14e5b0717503c73

Congratulations!