# **Mr Robot CTF**

## Flag 1

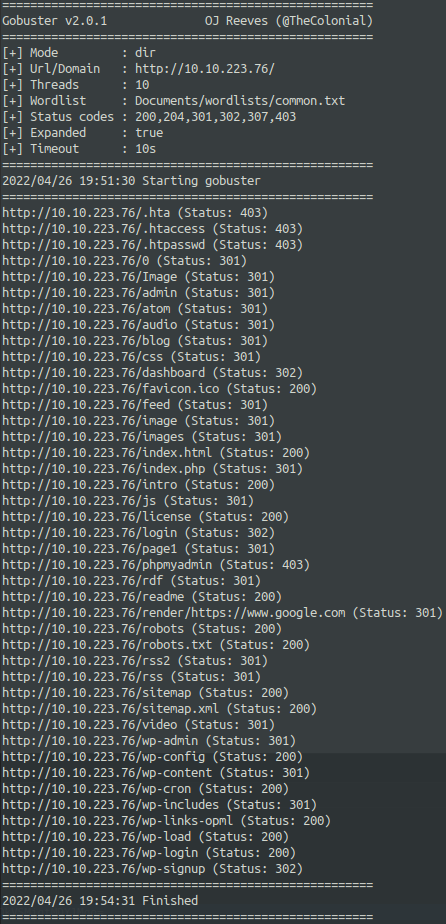
Context: Site robots.txt file.

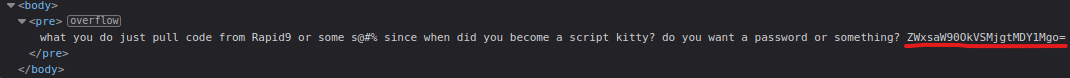
* If we open the site and navigate to the /robots.txt file, we will find some additional files to explore. Part of those files includes the first flag.

## Flag 2

Context: File directory enumeration w/ stored credential access + WordPress template code injection + Reverse Shell w/ PHP.

* The site doesn’t offer much in terms of providing an attack vector. Because of this, we should look at enumerating for any endpoints that the site may contain. We do this using the following command:

*gobuster -e -u {target\_ip} -w common.txt*

* After directory enumeration, we can see some endpoints containing “wp-{name}”. If we navigate to one of these endpoints, we can see that the site is built with WordPress. We will keep this in mind for later.
* If we navigate to /login, we require a username and password. Before having to perform a brute-force attack with BurpSuite or Hydra, we should explore these other endpoints. The Hint states “White coloured font”.
* At some point, we should visit the /license endpoint. Immediately, we are given some text on the page, which isn’t of much use. However, if we view the source code of the site, we can see some Base64 encoded text.
* If we enter this encoded text into [www.base64decode.org](http://www.base64decode.org), are provided what looks like some login credentials. Sure enough, they are the credentials to the WordPress Administrator account.
* Digging around the WordPress CMS panel, we can find a tab called “Appearances”, and an “Editor” tab within this section. If we navigate to any of the .php pages, we can look at the source code for the page. In doing so however, we may also be able to modify the page to initiate a connection to our local machine.
* We use the code from php-reverse-shell.php file for this.
* After applying the changes, we setup a Netcat listener on our machine before navigating to the page we just modified. Afterwards, we then access the page. This should start the connection, which it does in this case.
* From here, we navigate to the primary directories as per normal when a reverse shell has been created.
* *cd /home/{user}*
* Text

  Description automatically generatedWe find some files of interest under the {user} folder.
* If we view the password.raw-md5 file, we can see a username and what looks like a hash. If we put the hash into an online hash identifier, or use other sources, we find that the hash is of MD5 type. From here, we would need to crack the hash using a wordlist.
* The final step is to actually switch user accounts to ‘robot’, but we can’t do that from the current shell that we have. To do so, we have to stabilise the shell. In this machine, it looks like the server already has Python installed. Because of this, we can spawn a pty shell.
* After stabilising the shell and running the su {user} command, we are now able to view the contents of the second flag.

## Flag 3

Concept: Access control mis-configurations + GTFObins

* This final flag requires us to gain root access to the system.
* If we try to run sudo at all, we are presented with a message detailing that the current user does not have access to run sudo. As a result of this, we have to find if the server administrator had accidentally left any breadcrumbs for us to use as leverage.
* There are a few ways we could go about this:

1. Manually search for what available programs are installed on the server.
2. Run the command: apt list --installed | grep {command\_of\_interest}
3. Run the command: find / -perm -u=s -type f 2> /dev/null

* Text

  Description automatically generatedRegardless of what method is used, we can use resources such as GTFOBins, ExploitDB, etc to search for the manipulation of mis-configured programs.
* As a result, we get Flag 3.