flag shop

发表于 2021-01-04 分类于 Challenge， 2019， SCTF， Web
Challenge | 2019 | SCTF | Web | flag shop

点击此处获得更好的阅读体验

题目考点

- *Ruby*
- *ERB*
- *SSTI*

解题思路

备注：Ruby 摸得少，搜了一个下午都没搜到 Ruby 的全局变量--后来结束了和出题人 evoA 师傅一聊才知道得用美元符号的全局变量，哭了。在这里也还是写写 WriteUp 记录下。

打开靶机，发现是这样一个页面。

看下页面源码，主要关注后面这一段，先获取信息，失败就去请求 auth。

auth 之后会得到一个 jwt token。看来之后的请求我们也得带上这个。

扫下敏感文件，有 robots.txt。

访问一下这个路径，是源码。

```
require 'sinatra'
require 'sinatra/cookies'
require 'sinatra/json'
require 'jwt'
require 'securerandom'
require 'erb'
set :public_folder, File.dirname(__FILE__) + '/static'
FLAGPRICE = 1000000000000000000000000000000
#ENV["SECRET"] = SecureRandom.hex(xx)
configure do
  enable :logging
  file = File.new(File.dirname(__FILE__) + '/../log/http.log',"a+")
  file.sync = true
  use Rack::CommonLogger, file
end
get "/" do
  redirect '/shop', 302
end
get "/filebak" do
  content_type :text
  erb IO.binread __FILE__
end
get "/api/auth" do
  payload = { uid: SecureRandom.uuid , jkl: 20}
  auth = JWT.encode payload,ENV["SECRET"] , 'HS256'
  cookies[:auth] = auth
end
get "/api/info" do
  islogin
  auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
  json({uid: auth[0]["uid"],jkl: auth[0]["jkl"]})
end
get "/shop" do
  erb :shop
end
get "/work" do
  islogin
  auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
  auth = auth[0]
  unless params[:SECRET].nil?
    if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
      puts ENV["FLAG"]
    end
  end
  if params[:do] == "#{params[:name][0,7]} is working" then
    auth["jkl"] = auth["jkl"].to_i + SecureRandom.random_number(10)
    auth = JWT.encode auth,ENV["SECRET"] , 'HS256'
    cookies[:auth] = auth
    ERB::new("<script>alert('#{params[:name][0,7]} working successfully!')</script>").result
  end
end
post "/shop" do
  islogin
  auth = JWT.decode cookies[:auth],ENV["SECRET"] , true, { algorithm: 'HS256' }
  if auth[0]["jkl"] < FLAGPRICE then
    json({title: "error",message: "no enough jkl"})
  else
    auth << {flag: ENV["FLAG"]}
    auth = JWT.encode auth,ENV["SECRET"] , 'HS256'
    cookies[:auth] = auth
    json({title: "success",message: "jkl is good thing"})
  end
end
def islogin
  if cookies[:auth].nil? then
    redirect to('/shop')
  end
end
```

可以看到/work 那里有 ERB 模板，还直接把可控参数 name 拼进去了，那么这里我们就可以传入一些构造过的参数，来达到我们的目的了。比如 `name=<%=1%>`，就会得 1。

继续看看源码，同时注意有这样一段意义不明的代码。似乎得传入 SECRET 参数。那么就一起带上。

```
unless params[:SECRET].nil?
    if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
      puts ENV["FLAG"]
    end
  end
```

对照 Ruby 全局变量表 ，不断 fuzz，发现$有东西，

回溯到源码看看

```
unless params[:SECRET].nil?
    if ENV["SECRET"].match("#{params[:SECRET].match(/[0-9a-z]+/)}")
      puts ENV["FLAG"]
    end
  end
```

其在模板渲染之前之前有个匹配，就是这里。要是 SECRET 参数存在则对其进行匹配，用传入的这个值去和 `ENV["SECRET"]` 匹配，匹配上了就往终端输出 FLAG。意义不明的代码，但这里既然有匹配，就可以用全局变量读出来了，也就是用 `$` 来读取匹配前的内容。 那么这里读出来的就是 ENV 的 SECRET 的一部分了。 然后我们 SECRET 不传试试，这样括号里的匹配就不进行，只进行括号外的 `ENV["SECRET"]` 的匹配，再用全局变量 `$` 就可以读出 `ENV["SECRET"]` 了。

拿到了 secret 之后，到 jwt.io 伪造一下 cookie 里的 auth 里存的 jwt 令牌。jkl 设置为 `2000000000000000000000000000` 。

修改 cookie，买 flag。

然后再解析一下新的 jwt token。

Flag 到手~

- **本文作者：** CTFHub
- **本文链接：** https://writeup.ctfhub.com/Challenge/2019/SCTF/Web/gEhZwYqckPPXuYAWS4fuc3.html
- **版权声明：** 本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！

Challenge # Web # 2019 # SCTF
easyweb
math-is-fun1