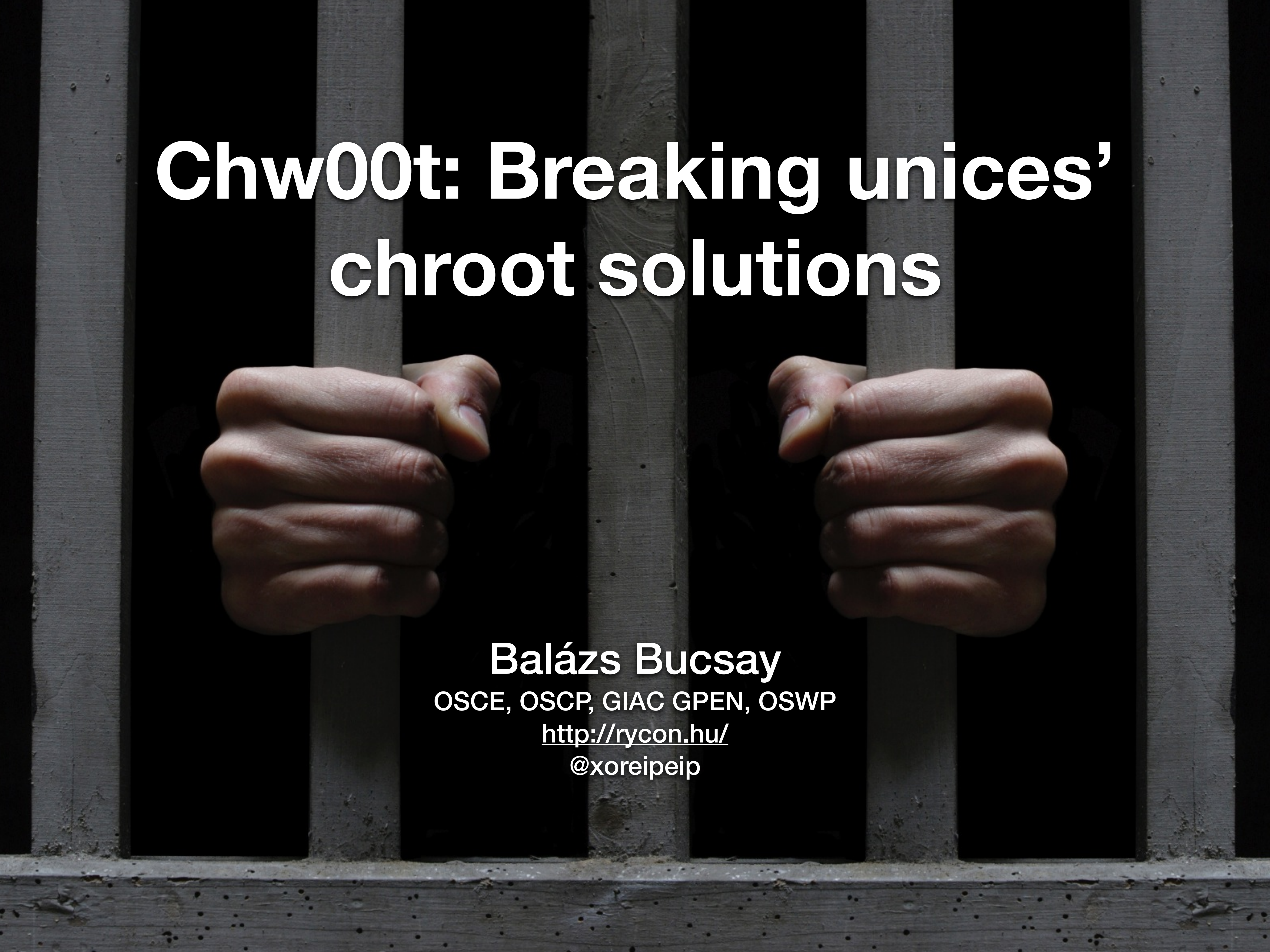


Chw00t: Breaking unices' chroot solutions

Balázs Bucsay
OSCE, OSCP, GIAC GPEN, OSWP
http://rycon.hu/
@xoreipeip

Bio / Balazs Bucsay

- Hungarian Hacker
- Strictly technical certificates: OSCE, OSCP, OSWP and GIAC GPEN
- Currently unemployed by choice - doing research
- Started with ring0 debuggers and disassemblers in 2000 (13 years old)
- Major project in 2009: GI John a distributed password cracker
- Presentations around the world (Atlanta, Moscow, London, Oslo)
- Webpage: http://rycon.hu
- Twitter: @xoreipeip
- Linkedin: http://www.linkedin.com/in/bucsayb

Chroot’s brief history

- Introduced in Version 7 Unix - 1979
- Inherited from V7 UNIX to BSD - 1982
- Hardened version was implemented in FreeBSD - 2000
- Virtuozzo (OpenVZ) containers - 2000
- Chroot on Steroids: Solaris container - 2005
- LXC: Linux Containers - 2008

What is Chroot?

- A privileged system call on Unix systems
- Changes the dedicated root vnode of a process (all children inherit this)
- Some OS stores chroots in linked lists
- Prevents access to outside of the new root
- Requires root: prevents crafted chroots for privilege escalation

What’s this used for?

- Testing environments
- Dependency control
- Compatibility
- Recovery
- Privilege separation??

LET ME ASK SOMETHING
IS THIS A SECURITY FEATURE?
memegenerator.net

NOPE

Requirements for reasonable chroot

- All directories must be root:root owned
- Superuser process cannot be run in chroot
- Distinct and unique user (uid, gid) has to be used
- No sensitive files (or files at all) can be modified or created

Requirements for reasonable chroot

- Close all file descriptors before chrooting
- chdir before chroot
- /proc should not be mounted
- + Use /var/empty for empty environment

Chroot scenarios

Shell access:

- SSH access to a chrooted environment
- Chrooted Apache running with mod_cgi/mod_php/…
- Exploiting a vulnerable chrooted app

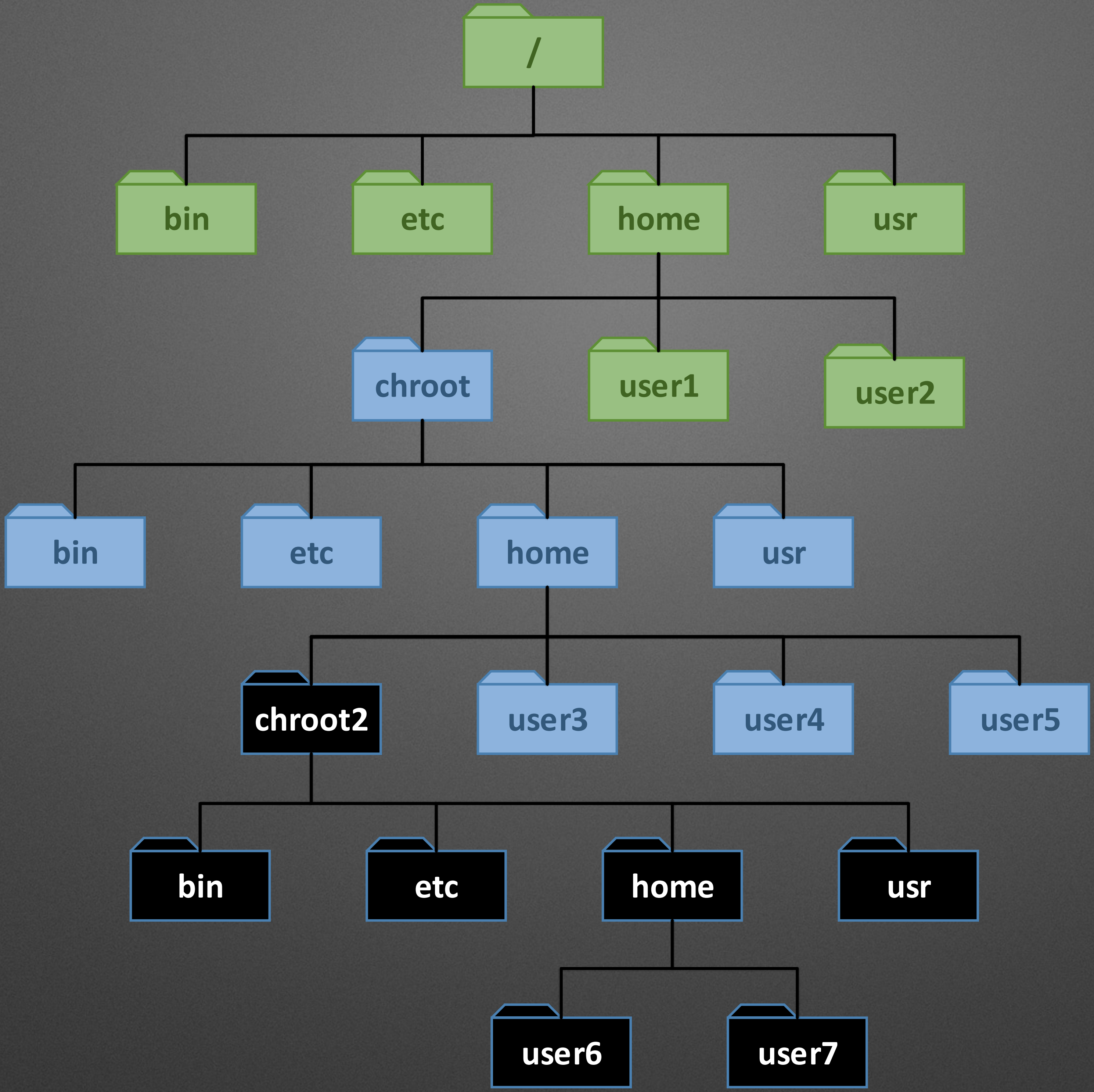
Only filesystem access:

- Chrooted SCP/FTP access

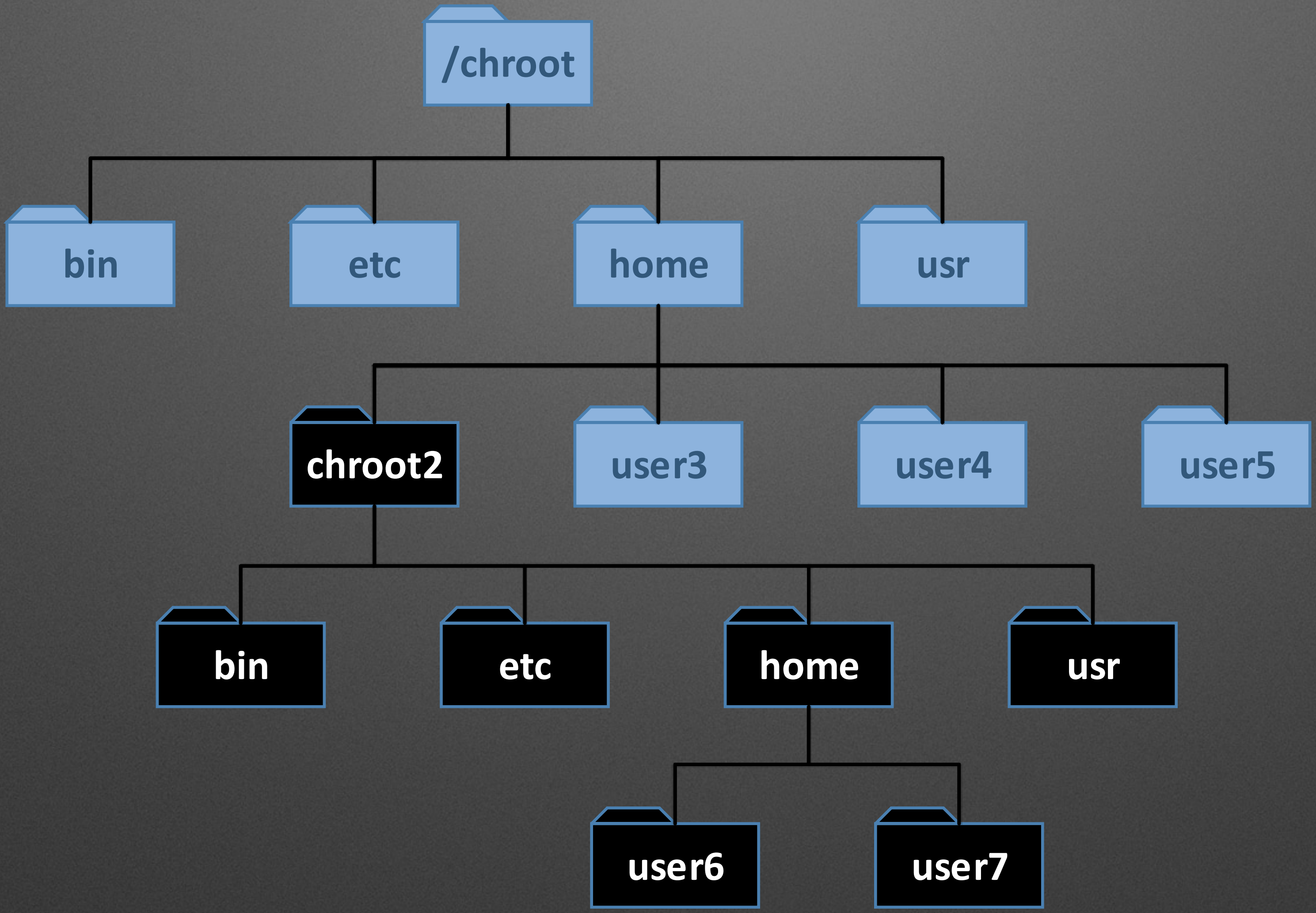
Breakage techniques

mostly summarised

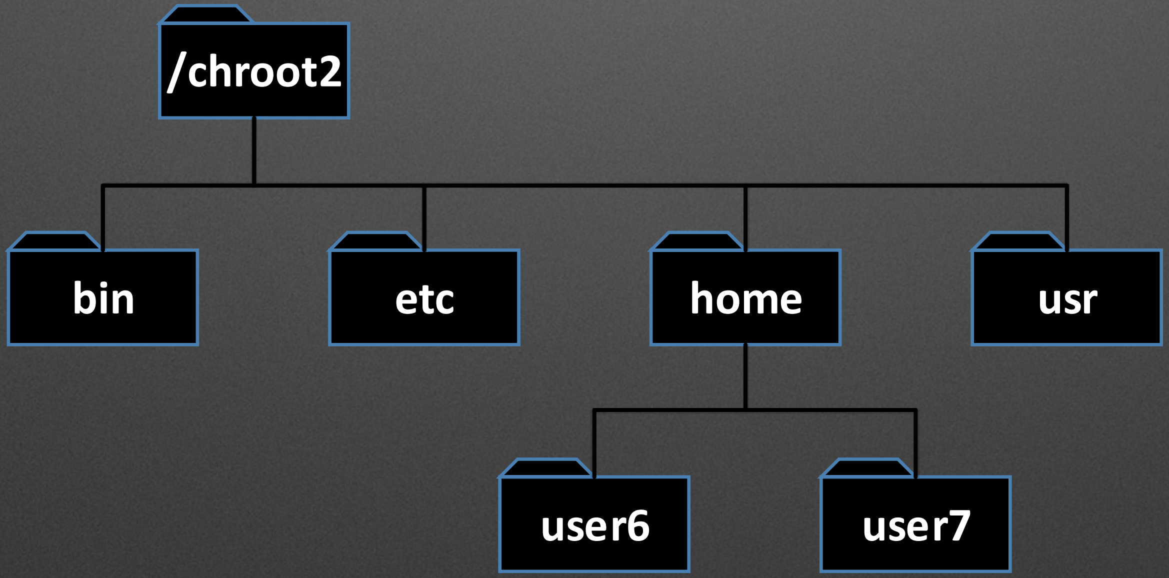
- Get root (not all techniques need it)
- Get access to a directory’s file descriptor outside of the chroot
- Find original root
- Chroot into that
- Escaped
- Only a few OS stores chroots in linked lists, if you can break out of one, you broke out all of them



Example structure
Original root



Example structure
New root (chrooted once)



Example structure
New root (chrooted twice)

Breakage techniques: kernel exploit/module

#root: MIGHT needed

Not going to talk about this

Breakage techniques: misconfigurations

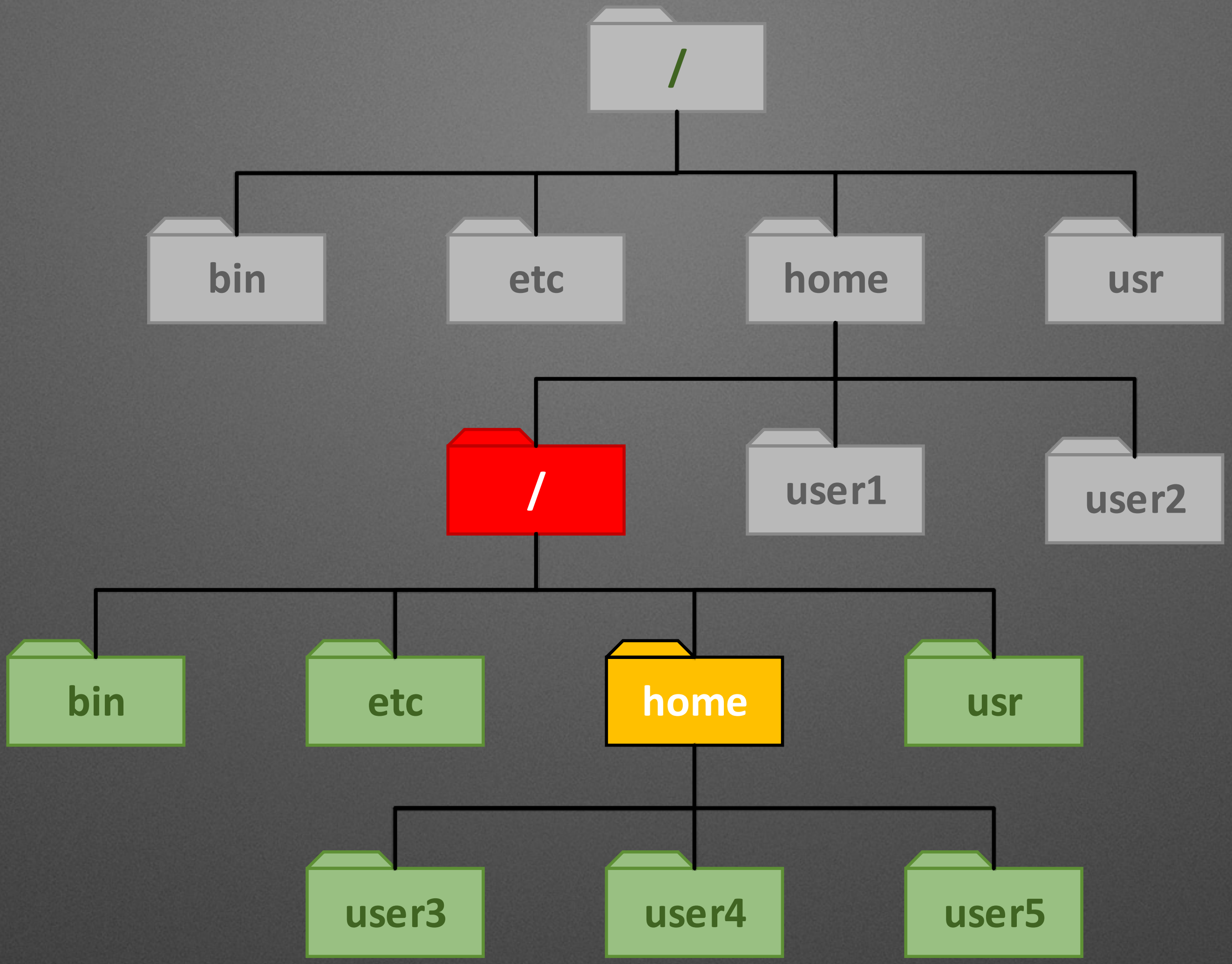
#root: NOT needed

- Hard to recognise and exploit
- Wrong permissions on files or directories
- Dynamic loading of shared libraries
- Hardlinked suid/sgid binaries using chrooted shared libraries
- For example:
  - /etc/passwd ; /etc/shadow
  - /lib/libpam.so.0 - used by /bin/su
- These can be used to run code as root

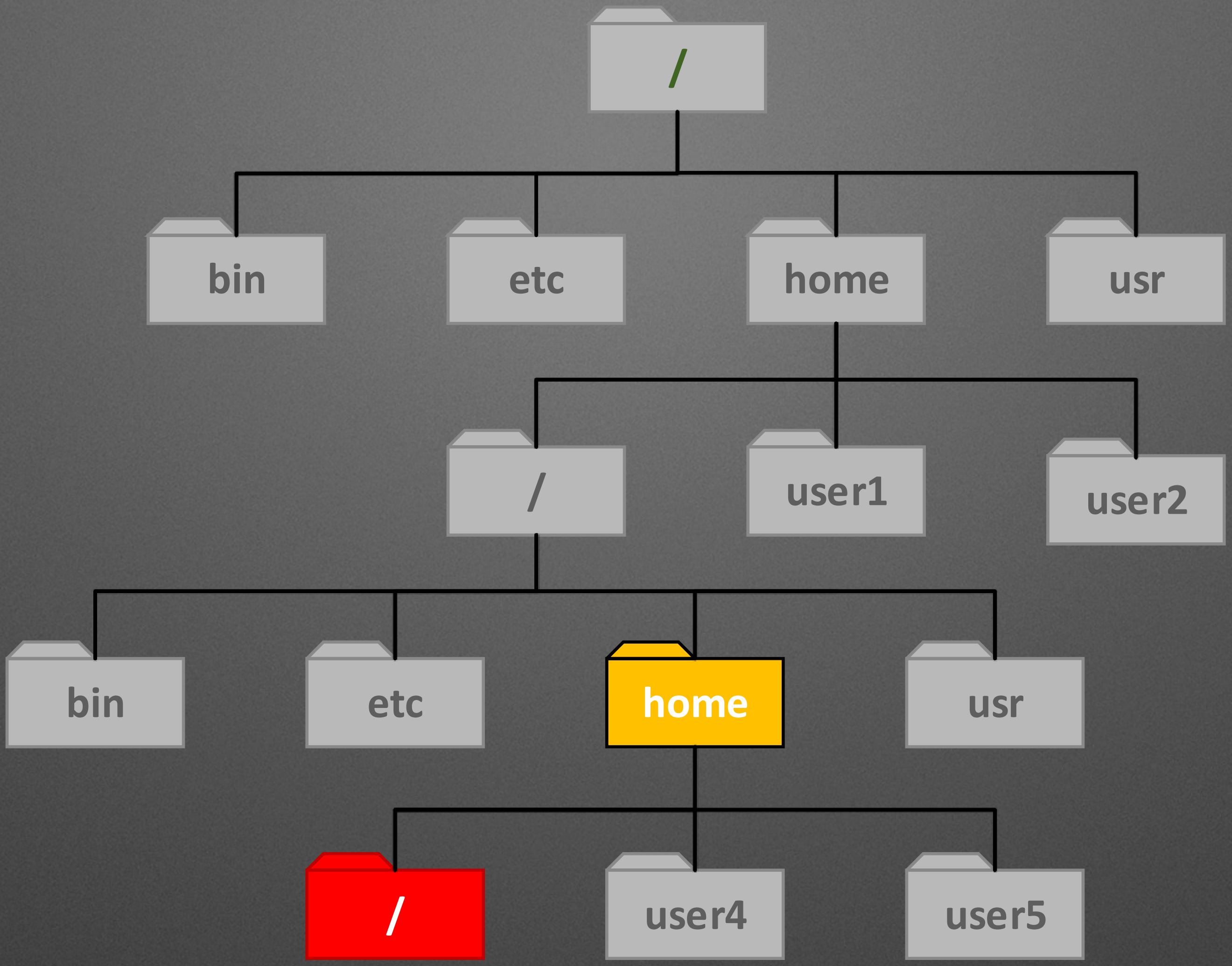
Breakage techniques: classic

#root: needed

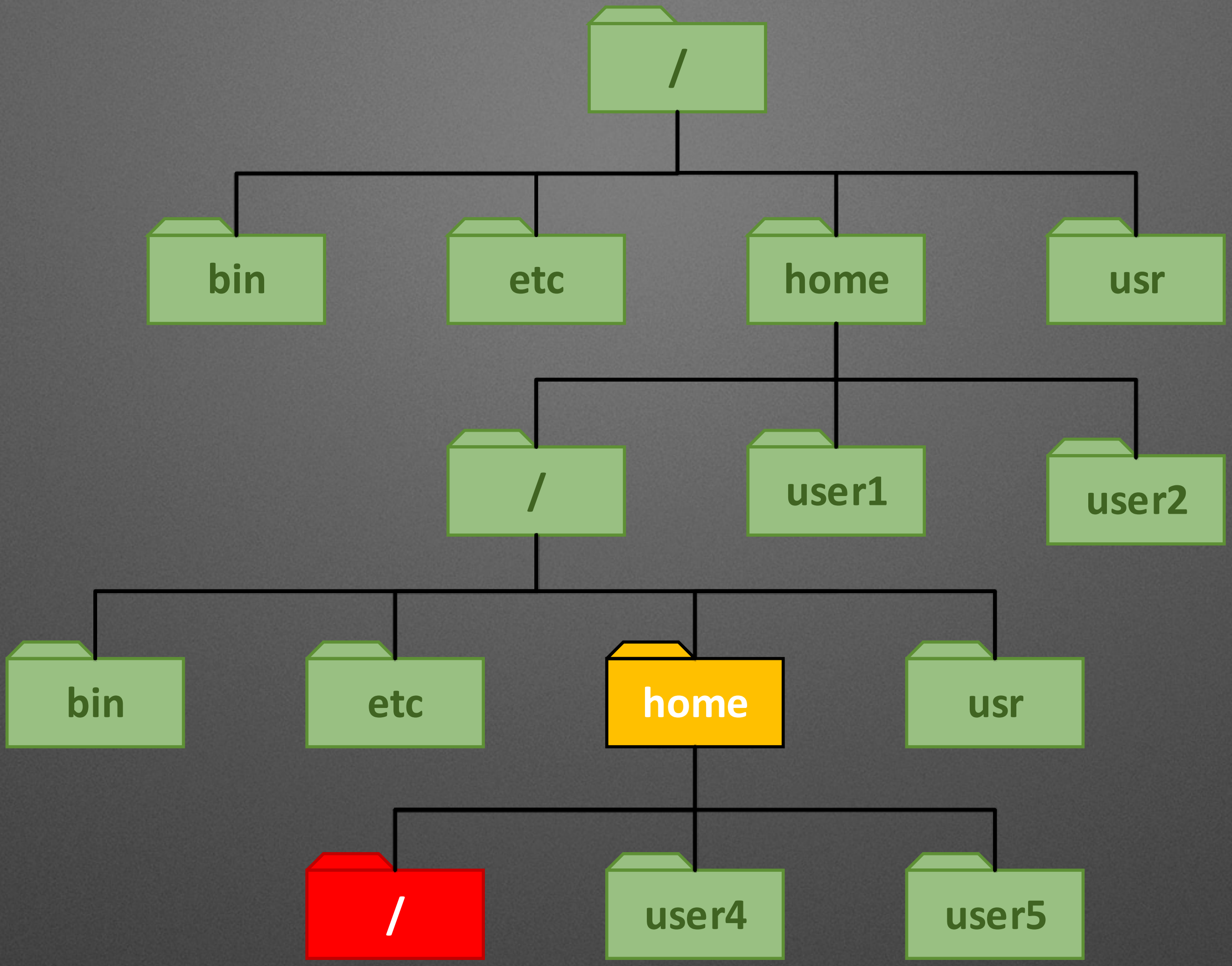
- Oldest and most trivial
- mkdir(d); chroot(d); cd ../../../; chroot(.)
- chroot syscall does not chdir into the directory, stays outside


/
bin
etc
home
usr
/
user1
user2
bin
etc
home
usr
user3
user4
user5

Root and CWD



Root barrier and CWD

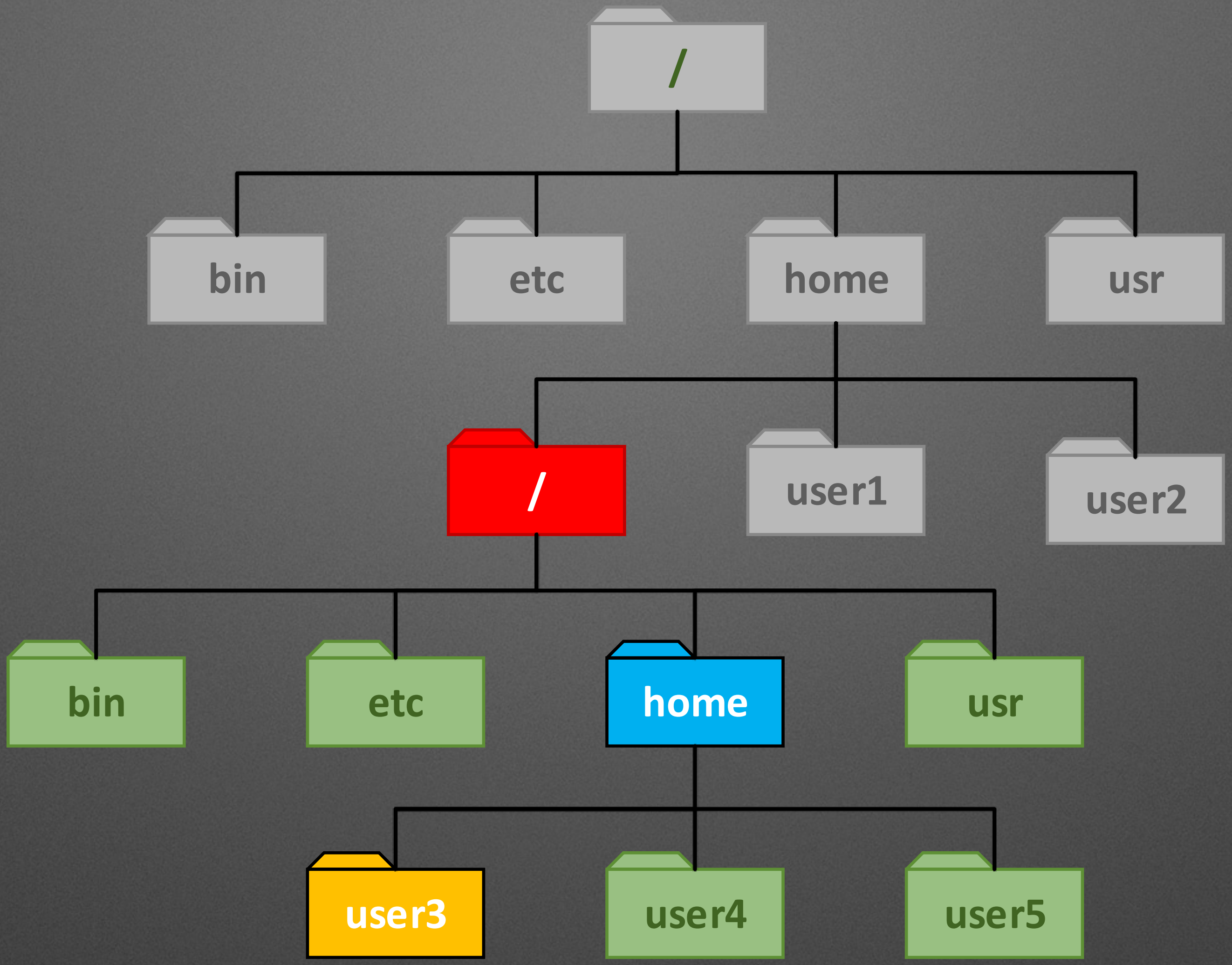


Root barrier and CWD

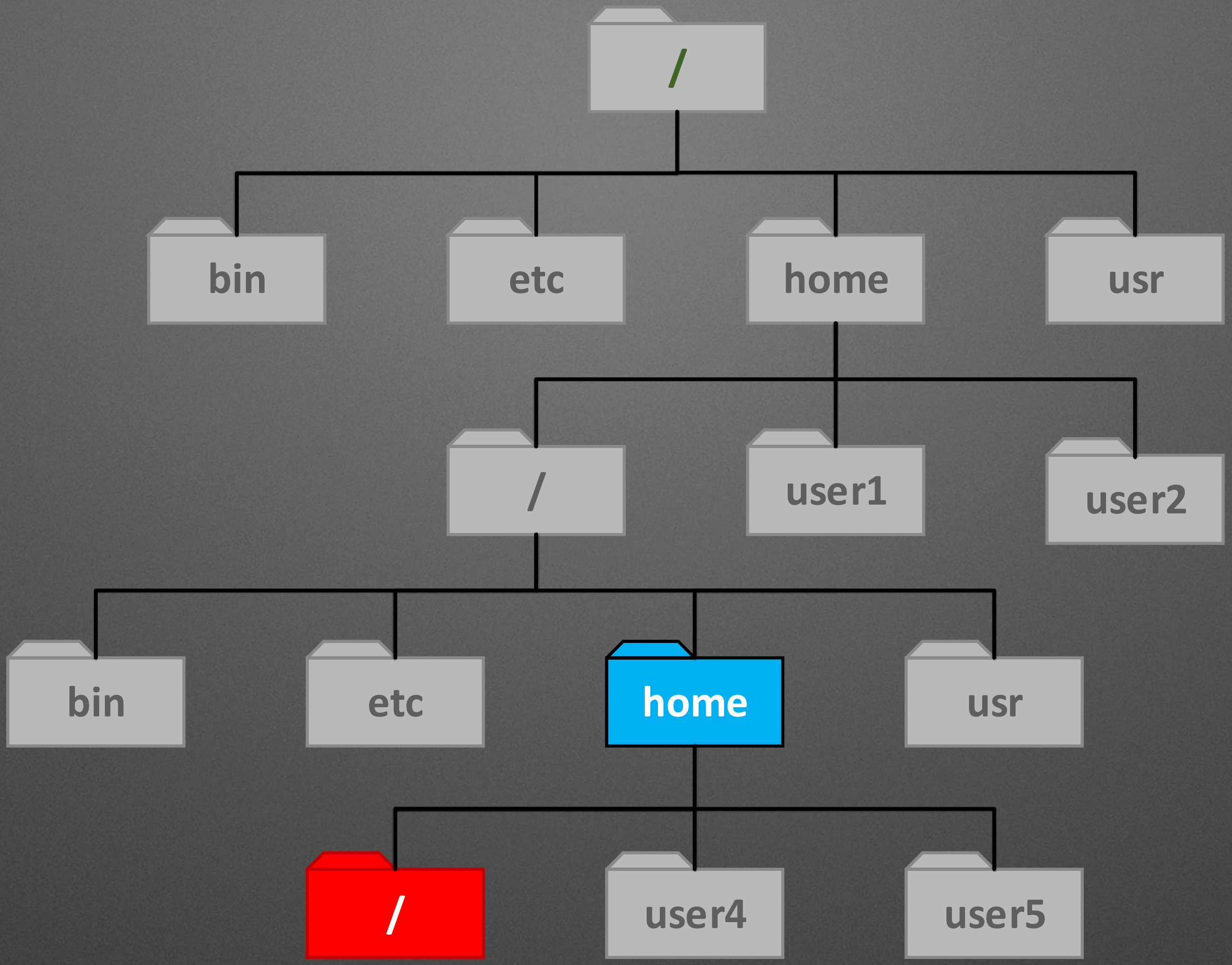
Breakage techniques: classic+fd saving

#root: needed

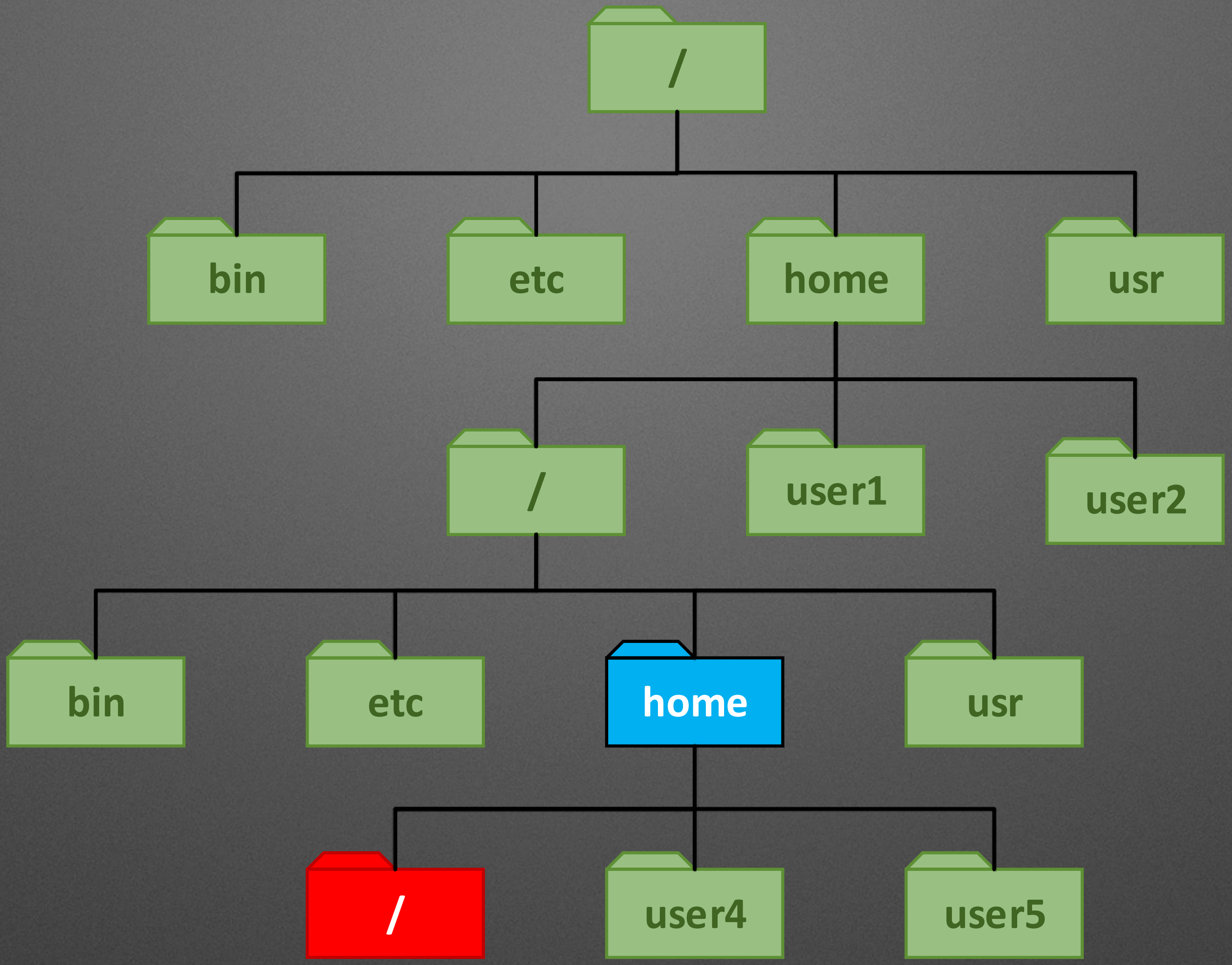
- Based on the classic
- Saving the file descriptor of CWD before chroot
- mkdir(d); n=open(.); chroot(d); fchdir(n); cd ../../../../; chroot(.)
- Some OS might change the CWD to the chrooted one



Root, CWD and saved fd



Root barrier and saved fd

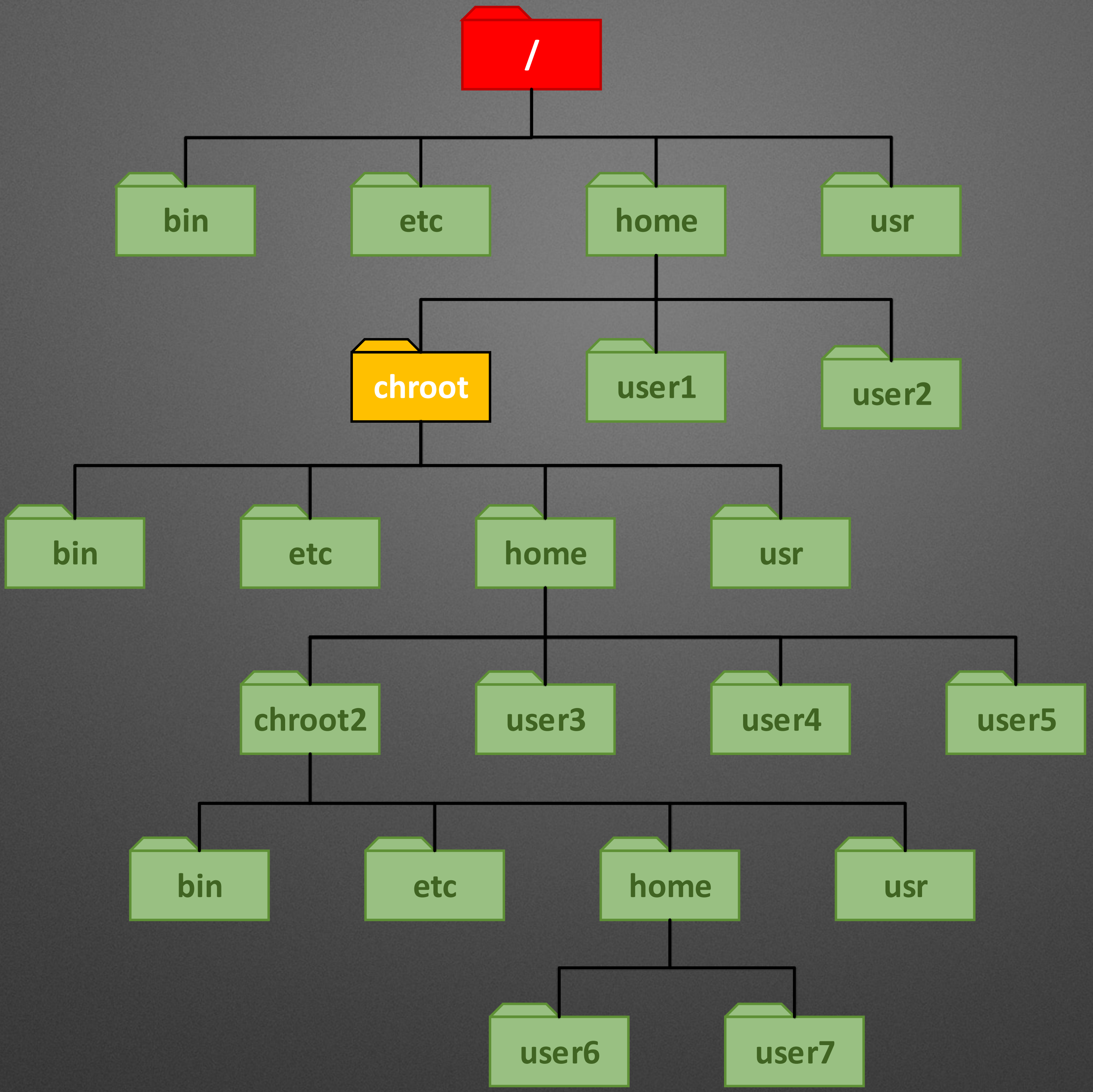


Root barrier and saved fd

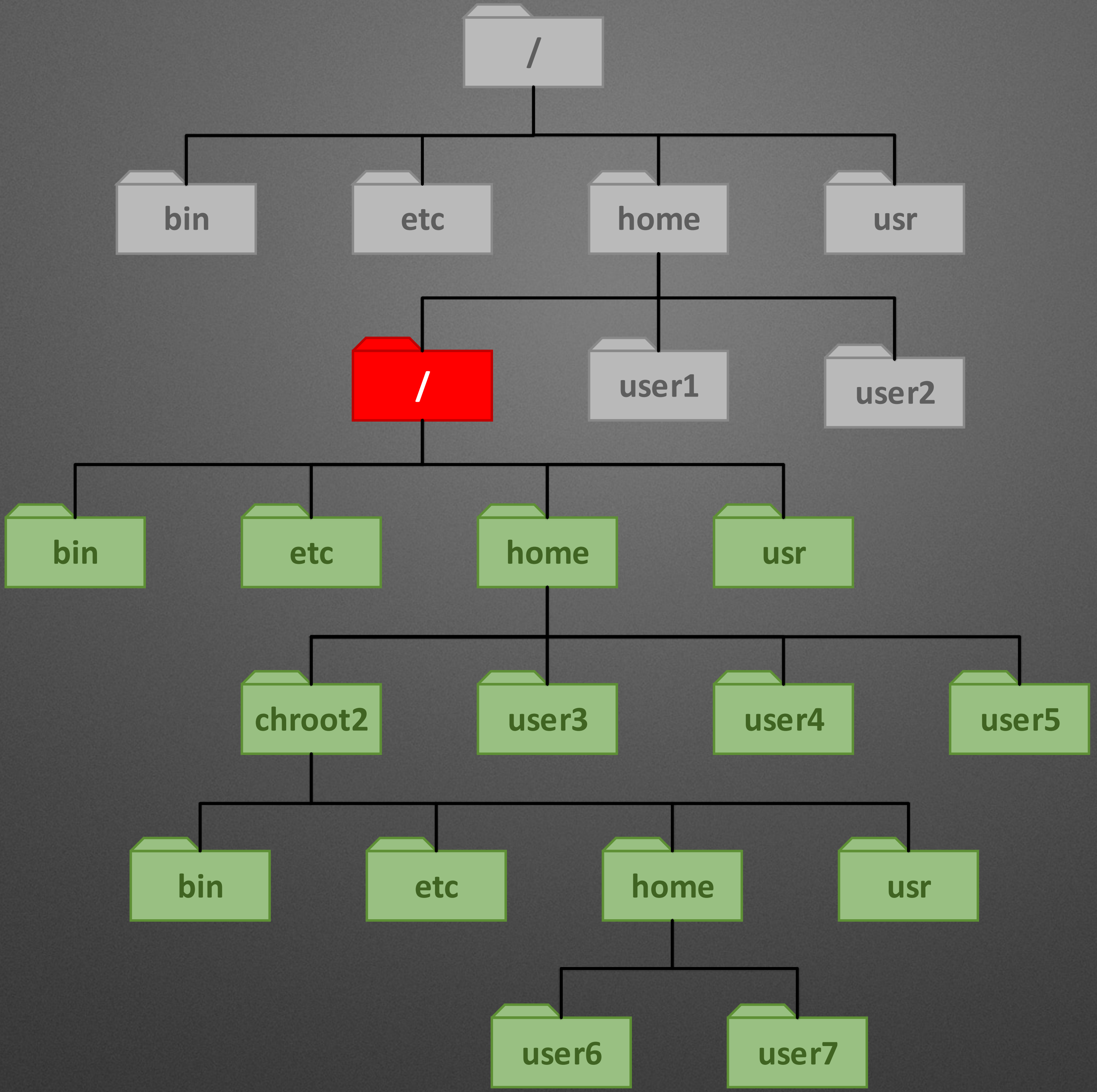
Breakage techniques: Unix Domain Sockets

#root: needed

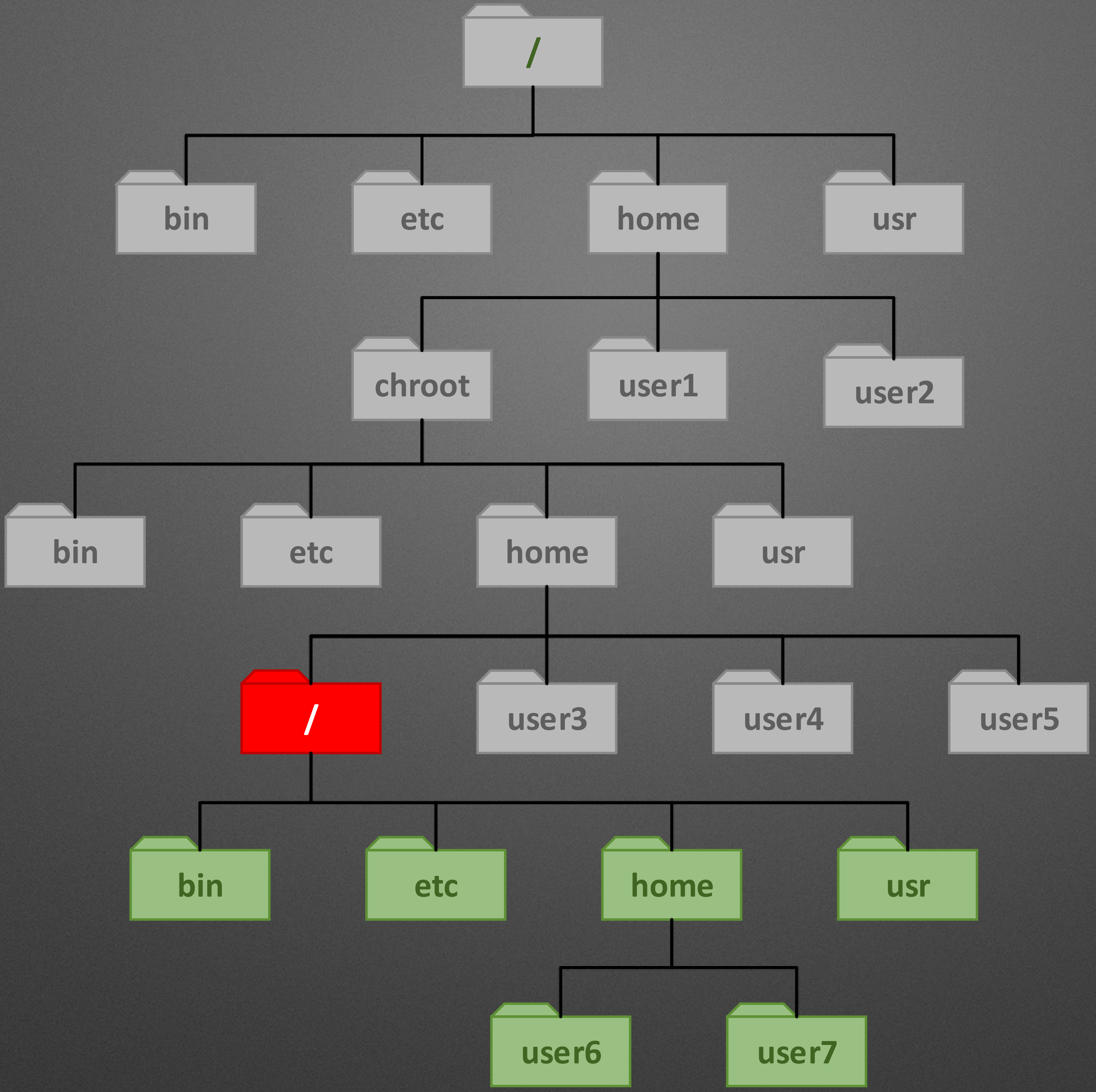
- UDS are similar to Internet sockets
- File descriptors can be passed thru
- Creating secondary chroot and passing outside fd thru
- Or using outside help (not really realistic)
- Abstract UDS does not require filesystem access



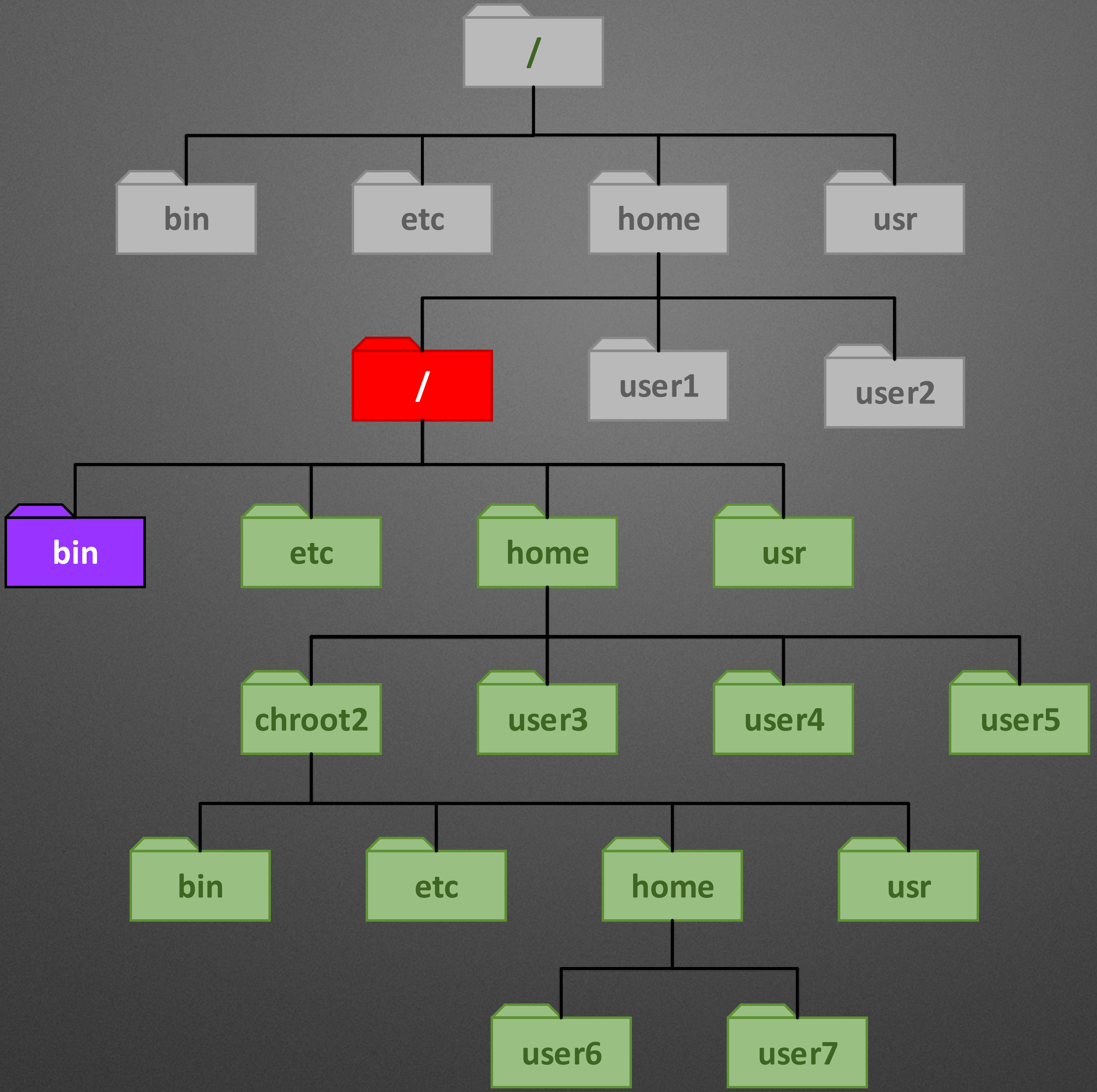
Root(0) and CWD



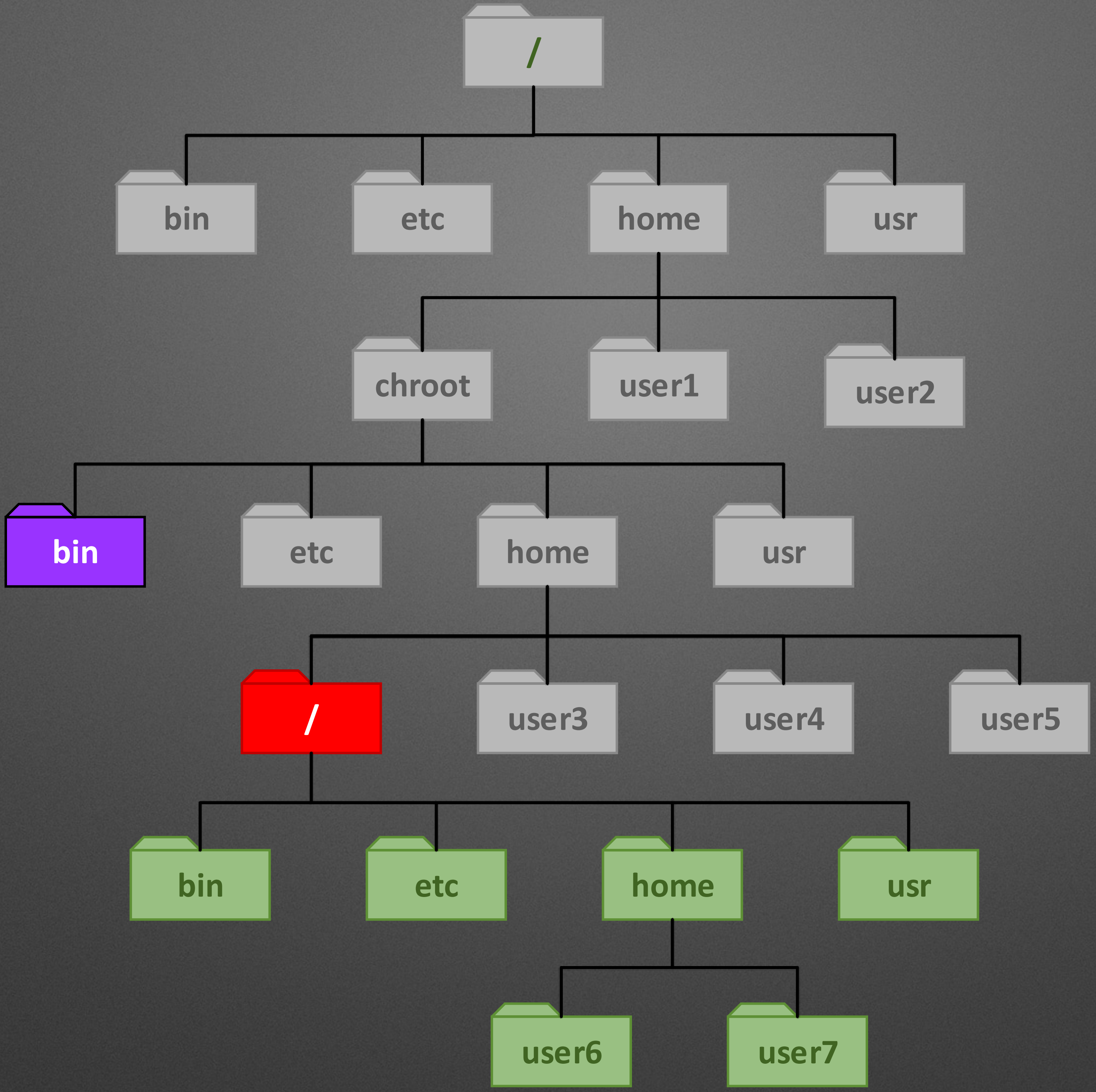
Root barrier(1) parent forks



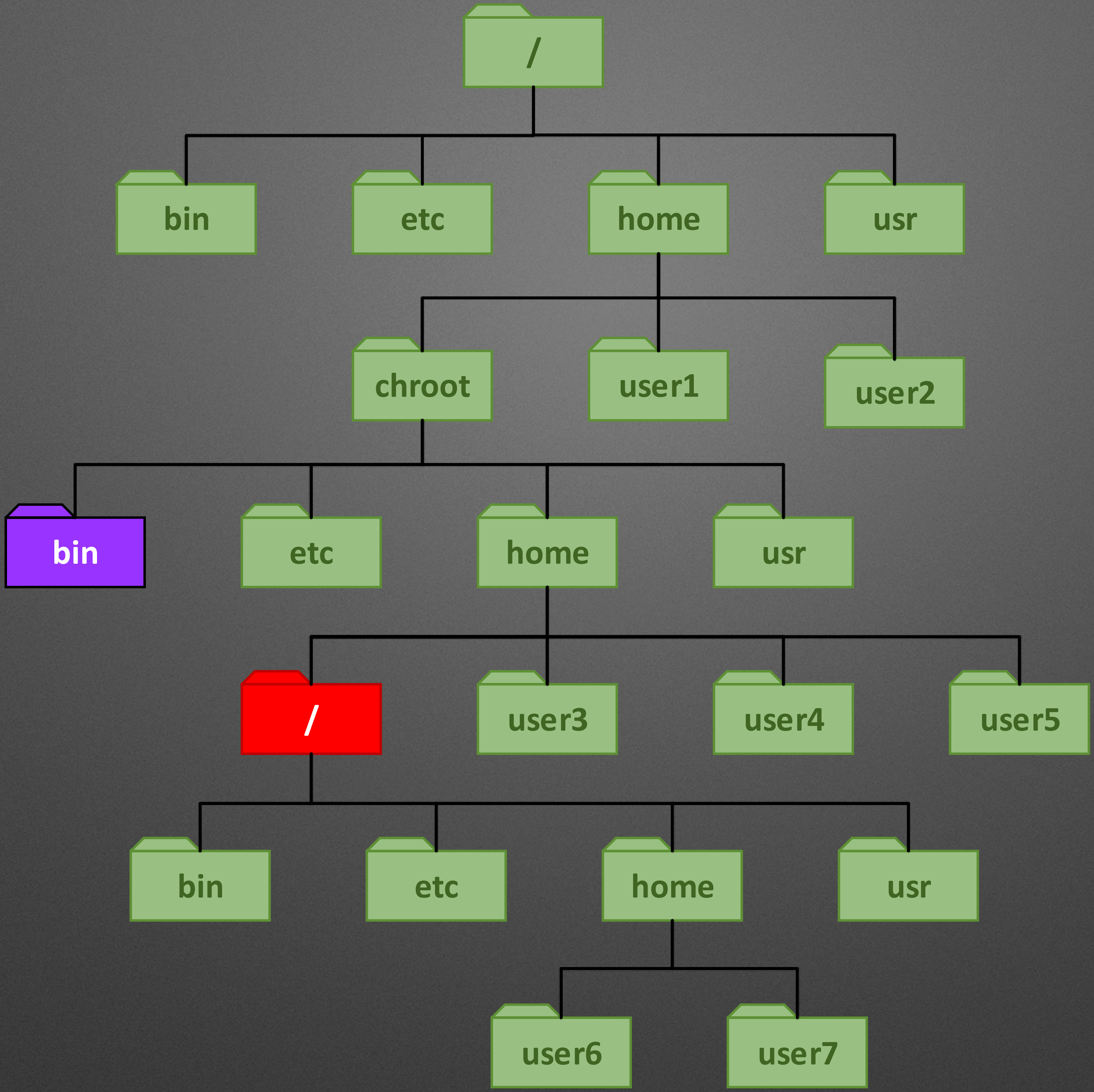
Root barrier(2) forked child



Root barrier(1) and FD (UDS)



Child Root barrier(2) and FD (UDS)



Child Root barrier(2) and FD (UDS)

Breakage techniques: mount()

#root: needed

- Mounting root device into a directory
- Chrooting into that directory
- Linux is not restrictive on mounting

Breakage techniques: /proc

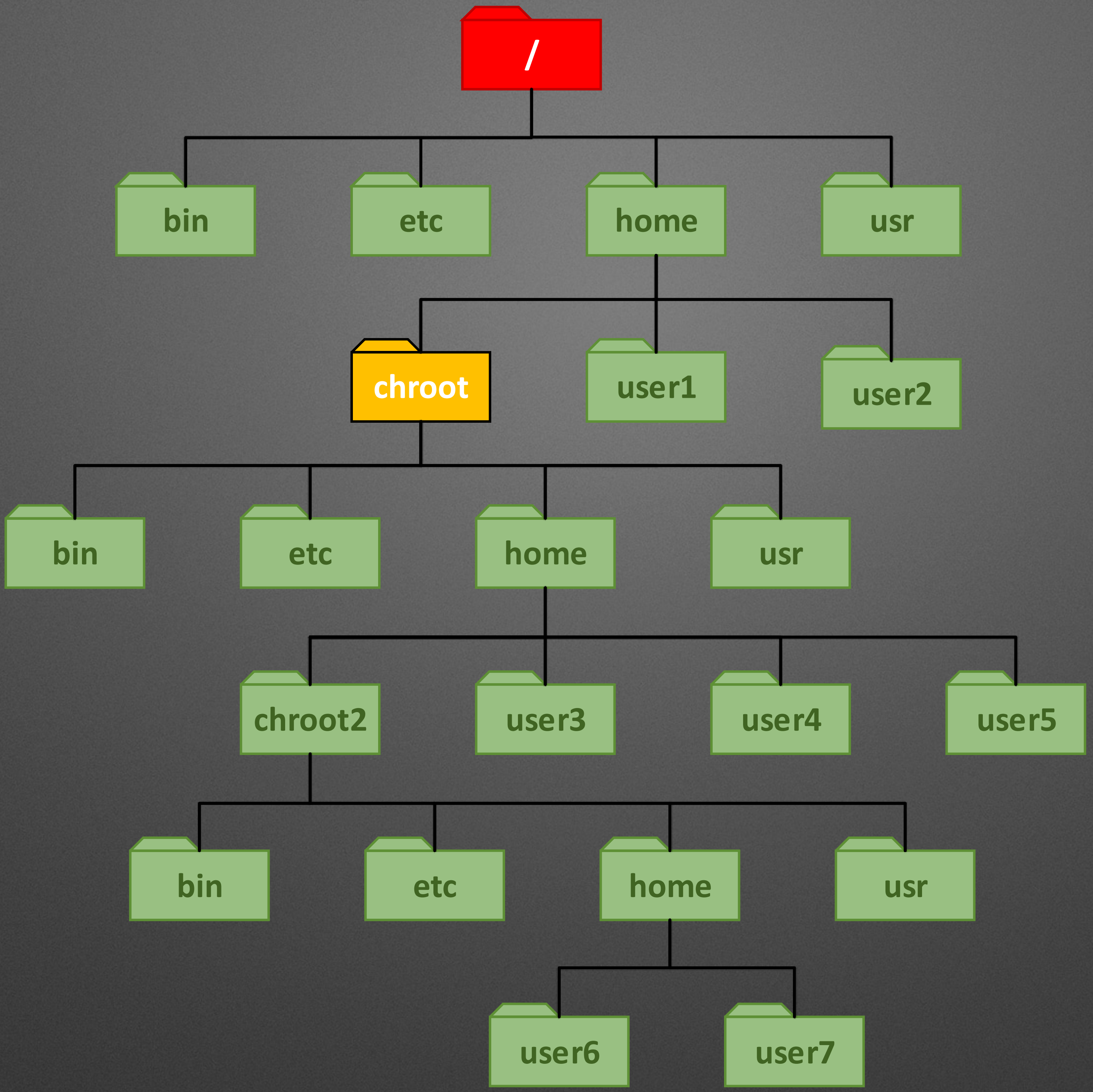
#root: needed

- Mounting procfs into a directory
- Looking for a pid that has a different root/cwd entry
- for example: /proc/1/root
- chroot into that entry

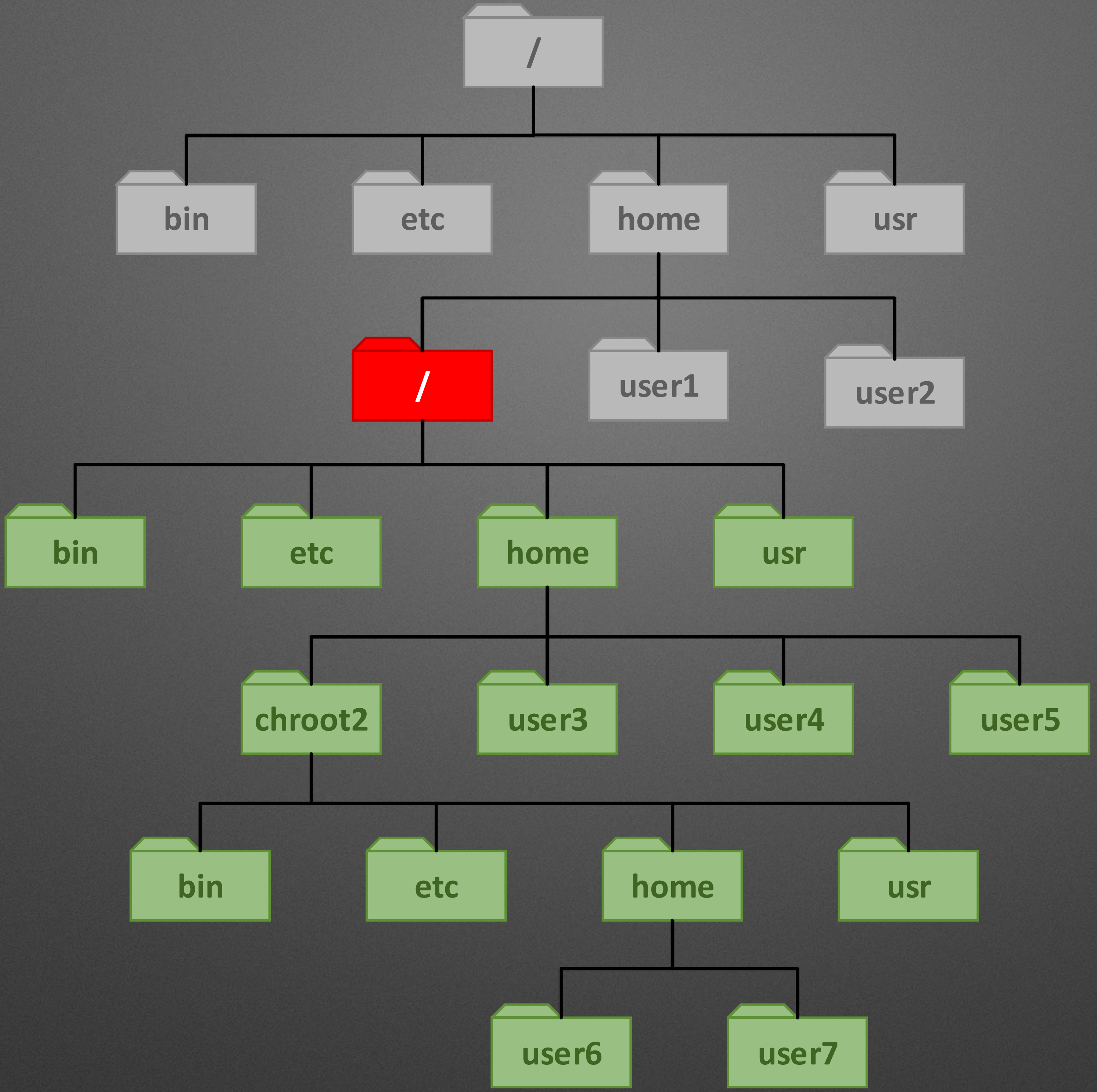
Breakage techniques: move-out-of-chroot

#root: MIGHT needed

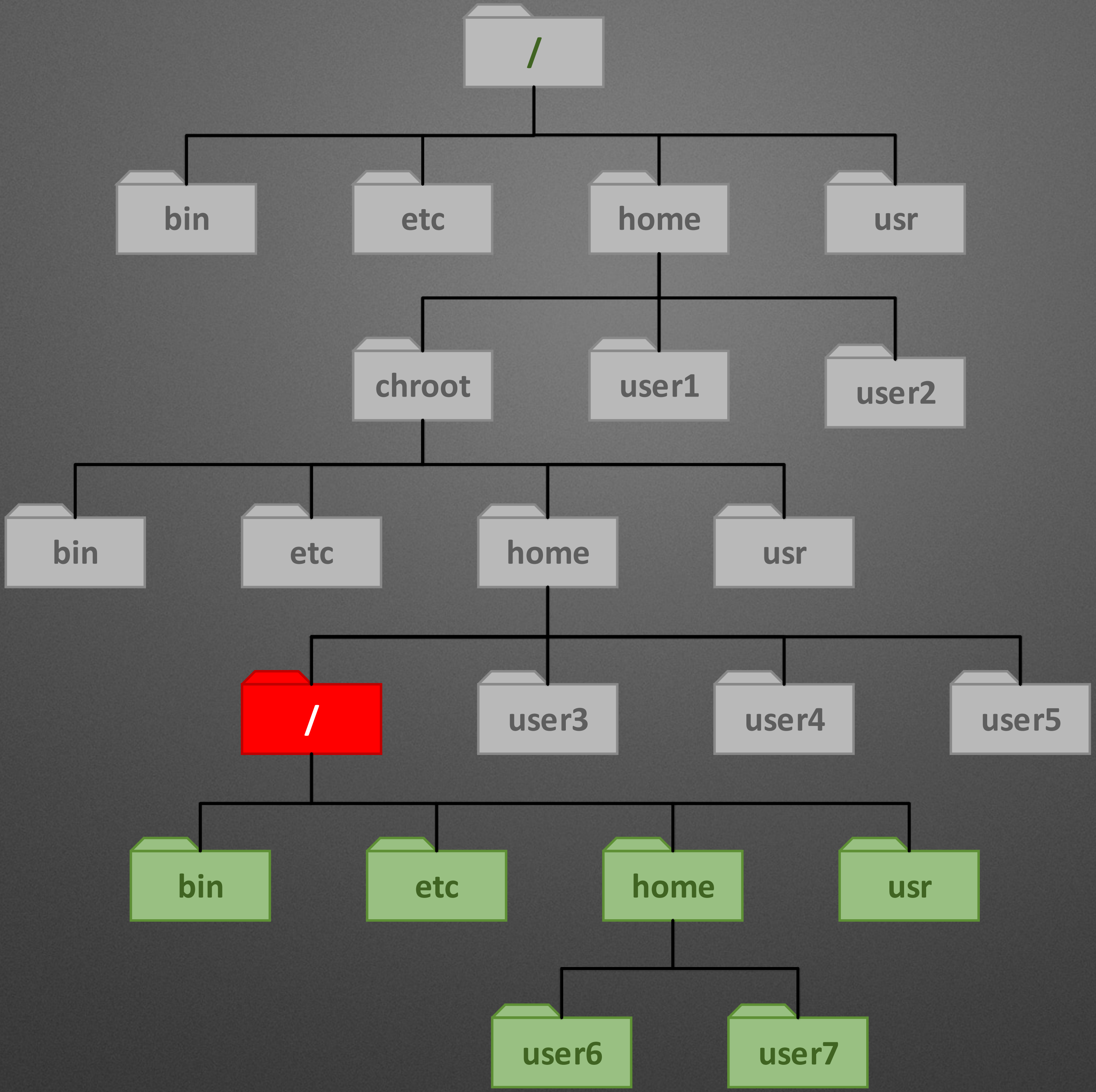
- The reason why I started to work on this
- Creating chroot and a directory in it
- Use the directory for CWD
- Move the directory out of the chroot



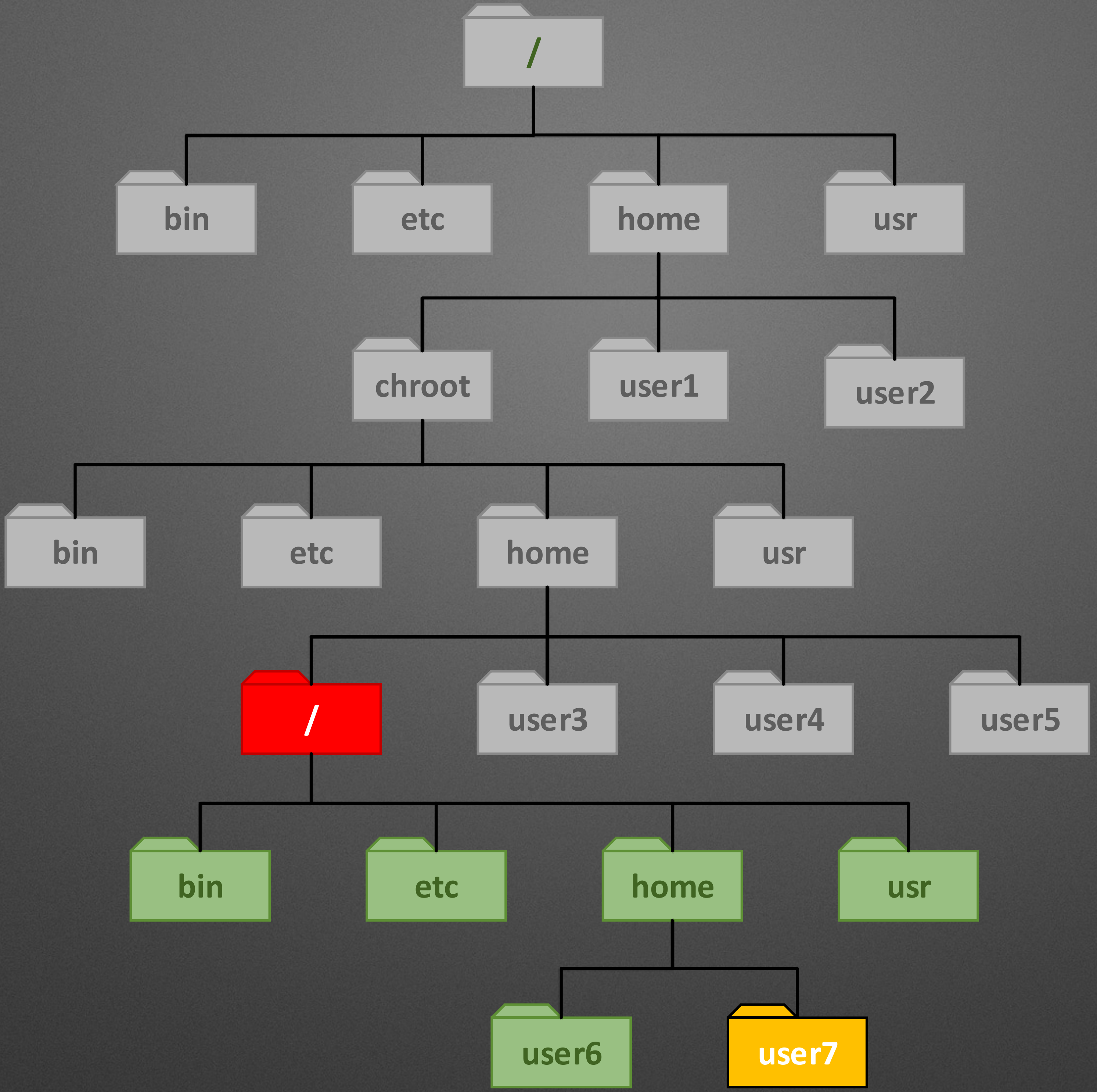
Root(0) and CWD



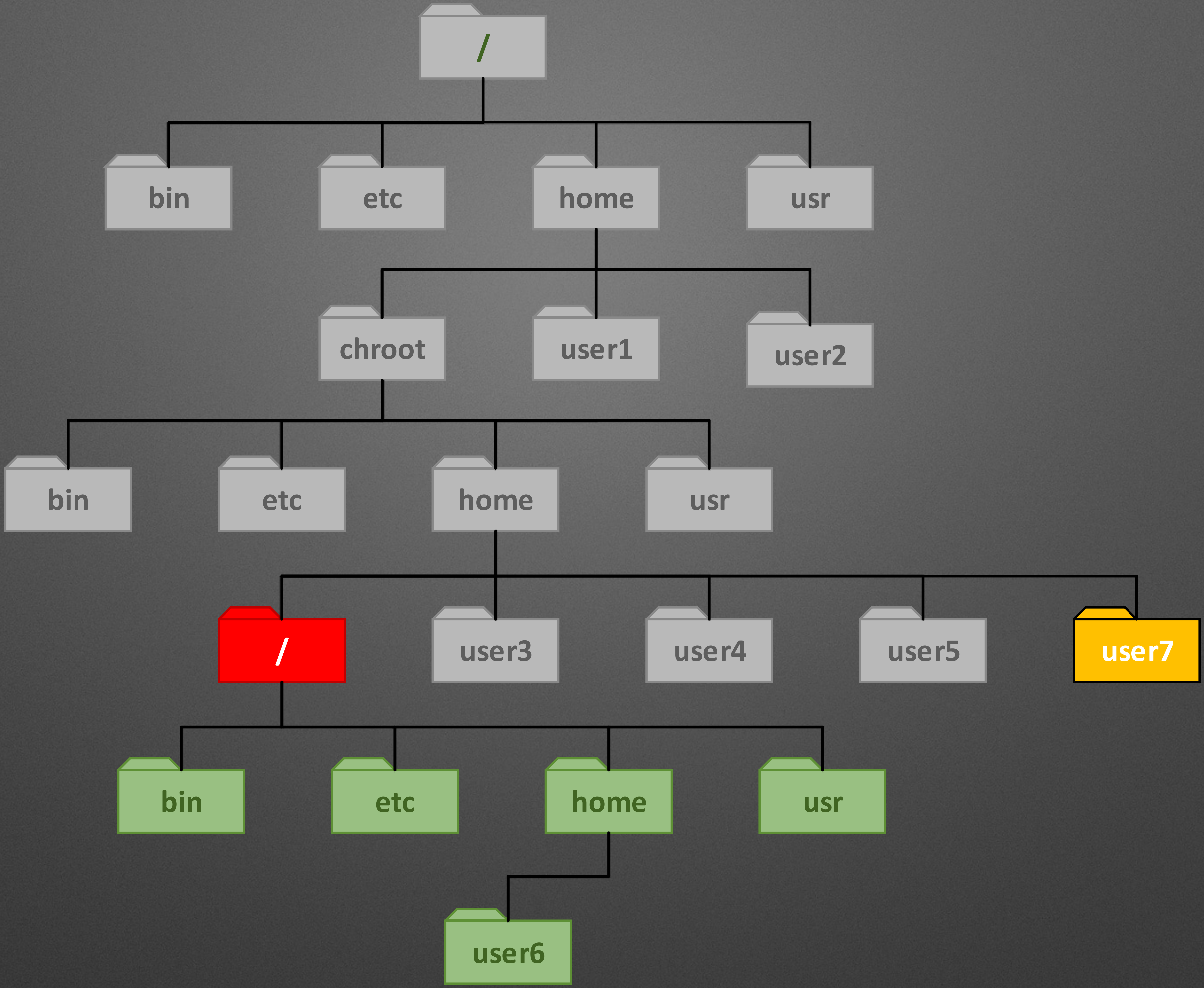
Root barrier(1) parent forks



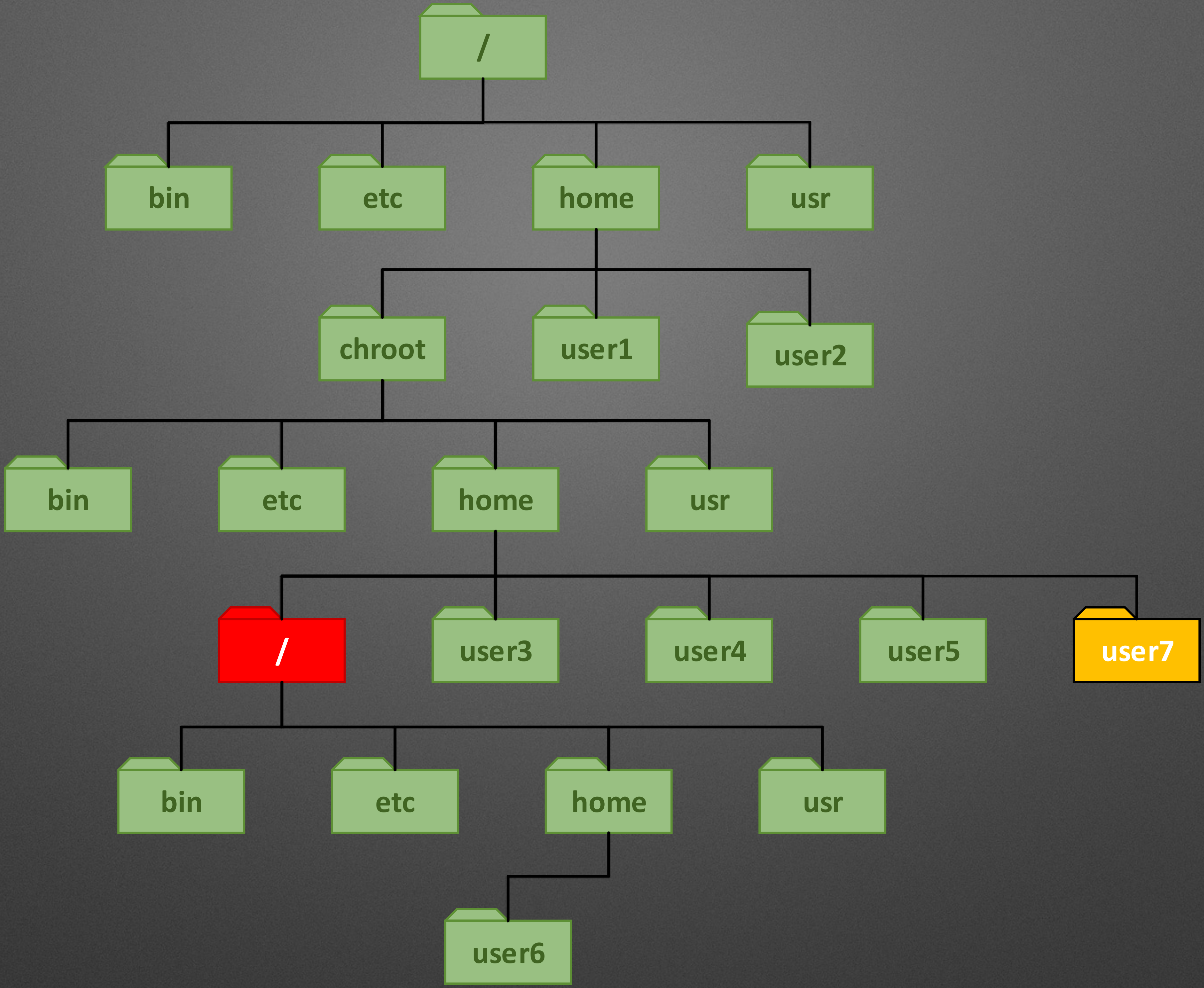
Root barrier(2) forked child



Root barrier(2) and CWD



Root barrier(2) and user7 moved out



Root barrier(2) and user7 moved out

Breakage techniques: ptrace()

#root: NOT needed

- System call to observe other processes
- Root can attach to any processes
- User can attach to same uid processes (when euid=uid)
- Change original code and run shellcode

Question

Tell me a service that is usually chrooted

DEMO

LIVE DEMO
I ALSO LIKE TO LIVE DANGEROUSLY
memegenerator.net

Results

	Debian 7.8;2.6.32/Kali 3.12	Ubuntu 14.04.1;3.13.0-32-generic	DragonFlyBSD 4.0.5 x86_64	FreeBSD 10.-RELEASE amd64	NetBSD 6.1.4 amd64	OpenBSD 5.5 amd64	Solaris 5.11 11.1 i386	Mac OS X
Classic	YES	YES	DoS	NO	NO	NO	YES	YES
Classic FD	YES	YES	NO	NO	NO	NO	YES	YES
Unix Domain Sockets	YES	YES	DoS	PARTIALLY	NO	PARTIALLY?	YES	YES
/proc	YES	YES	NO	NO	NO	NO	YES	NO
Mount	YES	YES	NO	NO	NO	NO	NO	NO
move out of chroot	YES	YES	DoS	PARTIALLY	NO	YES	YES	YES
Ptrace	YES	PARTIALLY	NO?	YES	NO	YES	N/A	N/A

Results (FreeBSD jail)

	FreeBSD 10. -RELEASE amd64	FreeBSD 10. Jail -RELEASE amd64
Classic	NO	NO
Classic FD	NO	NO
Unix Domain Sockets	PARTIALLY	PARTIALLY
Mount	NO	NO
/proc	NO	NO
move-out-of-chroot	PARTIALLY	PARTIALLY
Ptrace	YES	NO

Filesystem access only

- Move-out-of-chroot still works on FTP/SCP
- Privilege escalation is possible on misconfigured environment
- Shell can be popped by replacing or placing shared libraries/malicious files in chroot

Linux Containers

- Privileged container (no user namespaces) can create nested containers
- Host container has access to guest container's filesystem
- Based on the move-out-of-chroot technique, real host's file system is accessible

DEMO 2

LIVE DEMO
I ALSO LIKE TO LIVE DANGEROUSLY
memegenerator.net

Tool

https://www.github.com/earthquake/chw00t/

Future work

- Testing new UNIX operating systems (eg. AIX, HP-UX)
- Looking for other techniques

Future work

Greetz to:

- My girlfriend and family
- Wolphie and Solar Designer for mentoring
- Spender and Kristof Feiszt for reviewing

References

- http://www.bpfh.net/simes/computing/chroot-break.html
- http://www.unixwiz.net/techtips/chroot-practices.html
- http://linux-vserver.org/Secure_chroot_Barrier
- http://phrack.org/issues/59/12.html
- http://lwn.net/Articles/421933/
- https://securityblog.redhat.com/2013/03/27/is-chroot-a-security-feature/

Thank you

Q&A

http://rycon.hu

https://github.com/earthquake

@xoreipeip