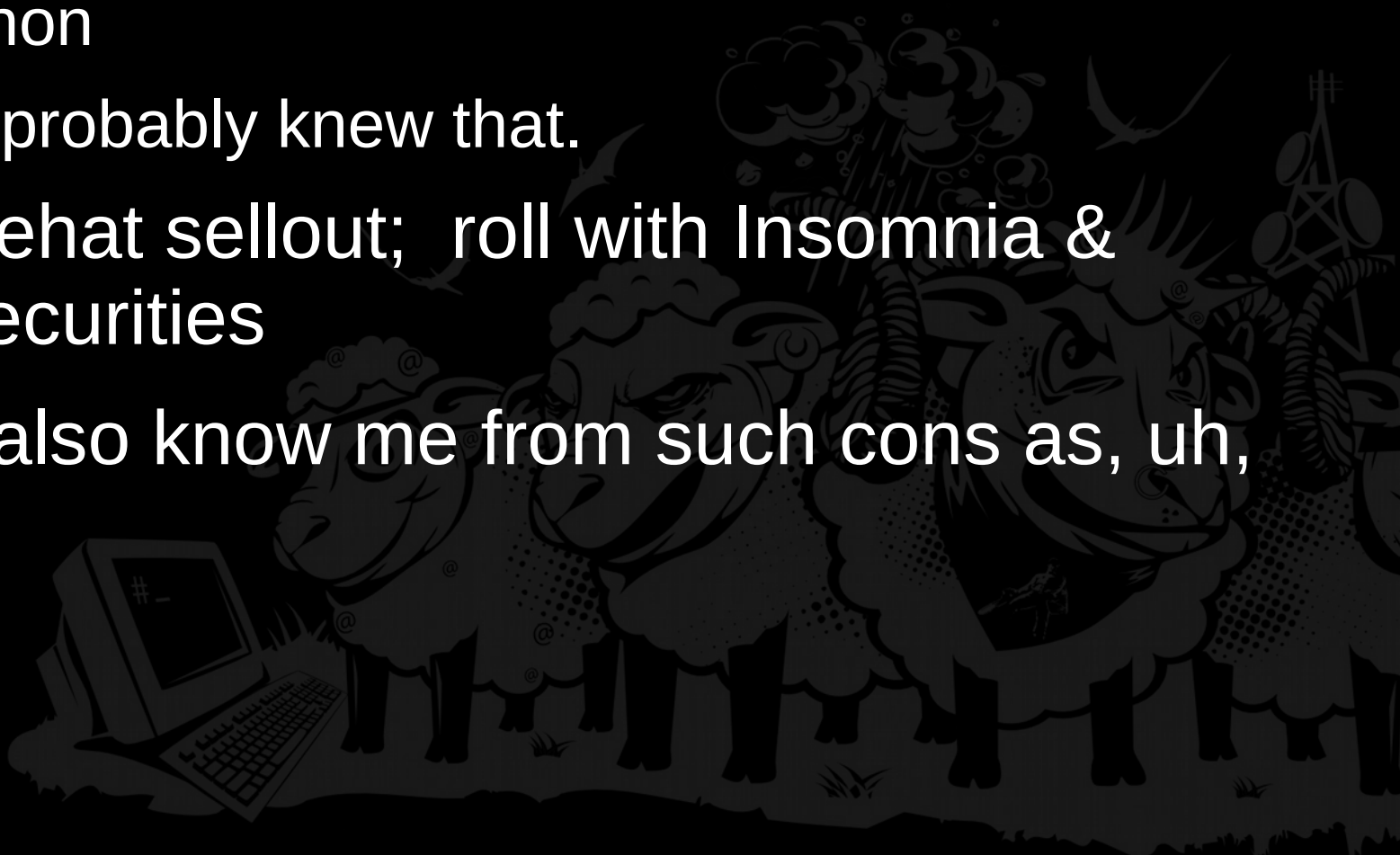
The Shell Game

Metlstorm
Kiwicon 4, Nov 2010

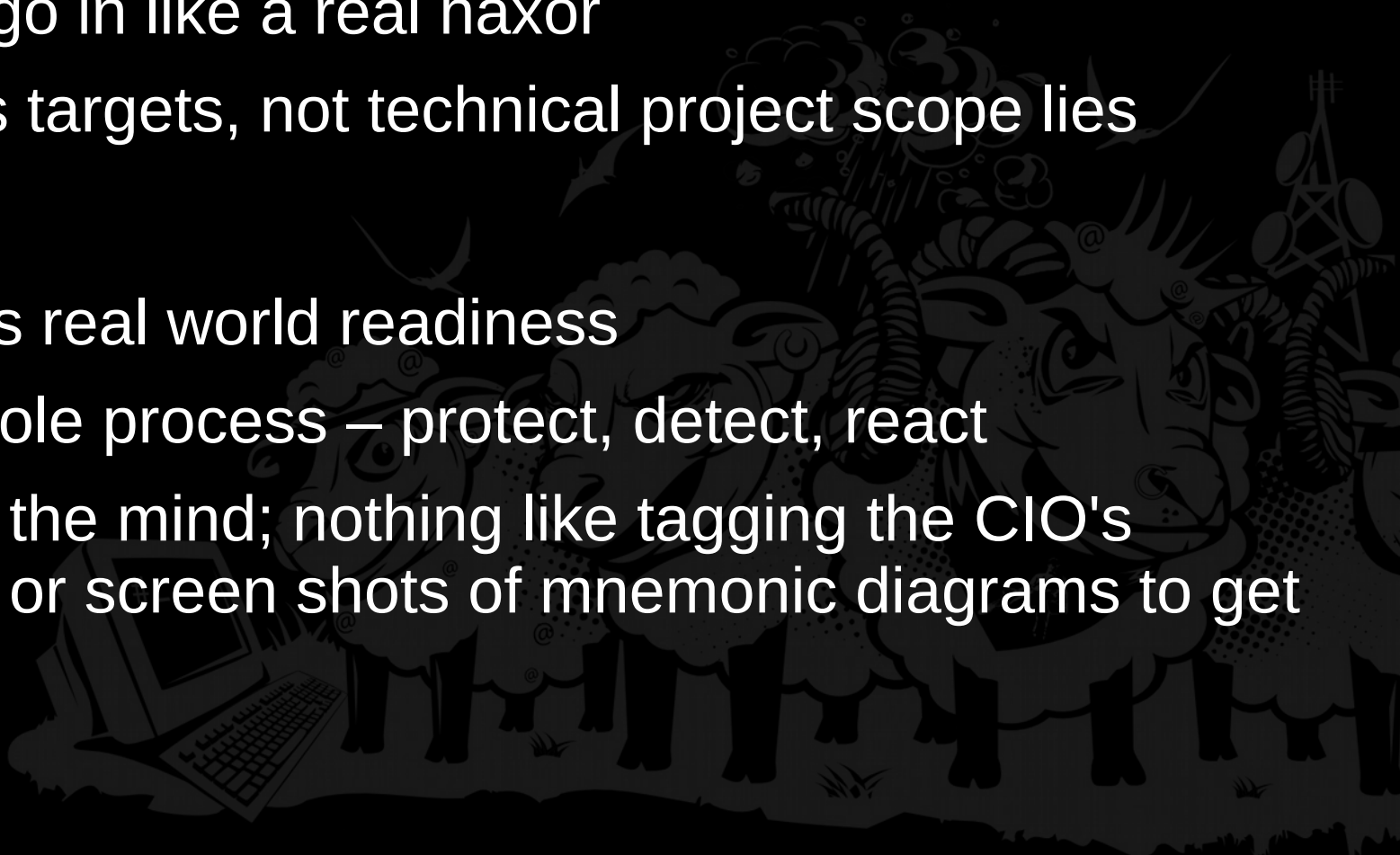
Intro

- Hi, I'm Metlstorm
  - I like unix
  - And python
  - But you probably knew that.
- I'm a whitehat sellout; roll with Insomnia & Lateral Securities
- You may also know me from such cons as, uh, Kiwicon.



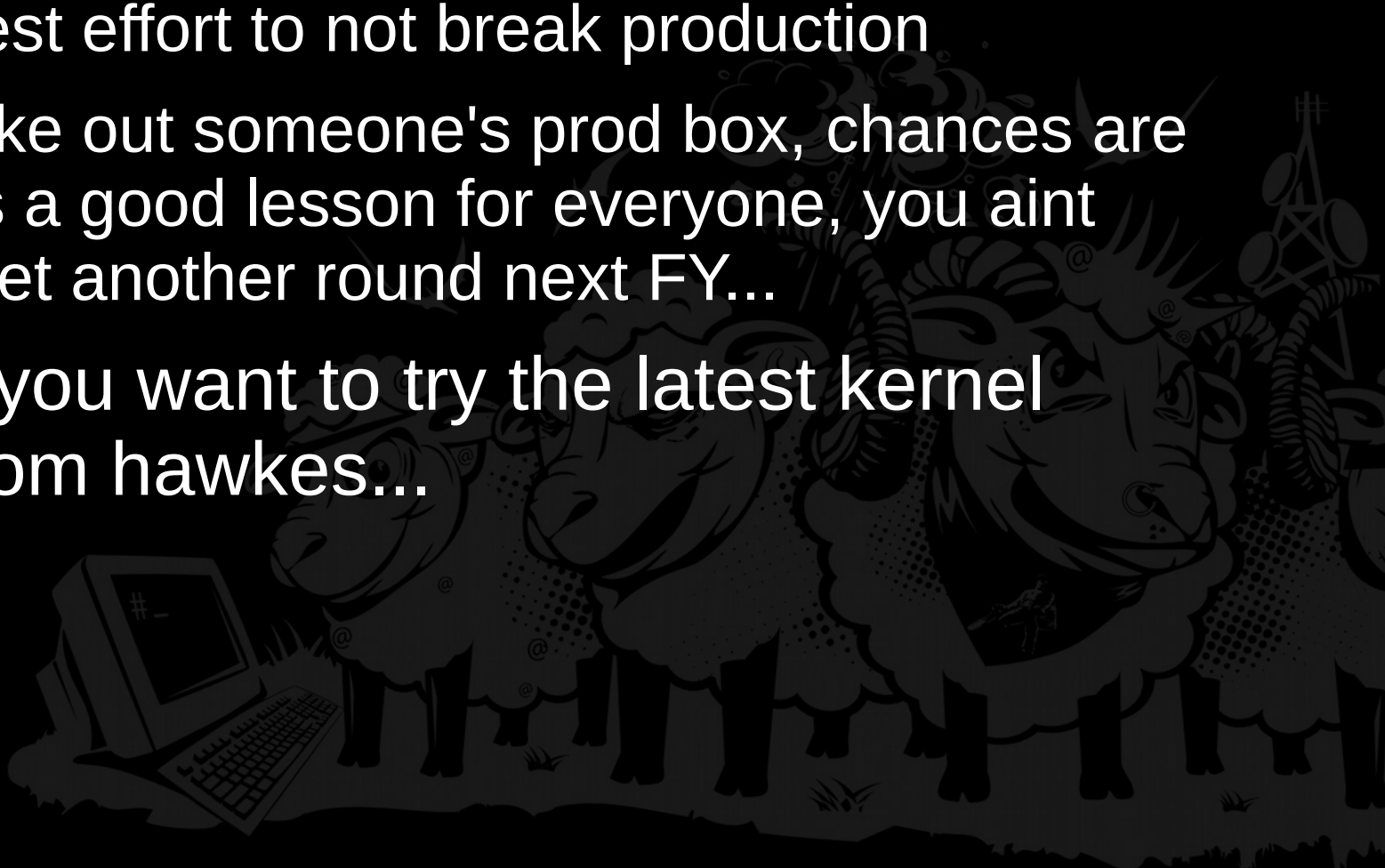
On Selling Out

- I do a fair bit of “Red Team” testing
  - Wide ranging, not much scope, external testing
  - No info, go in like a real haxor
  - Business targets, not technical project scope lies
- Value of
  - Illustrates real world readiness
  - Tests whole process – protect, detect, react
  - Focuses the mind; nothing like tagging the CIO's desktop or screen shots of mnemonic diagrams to get attention



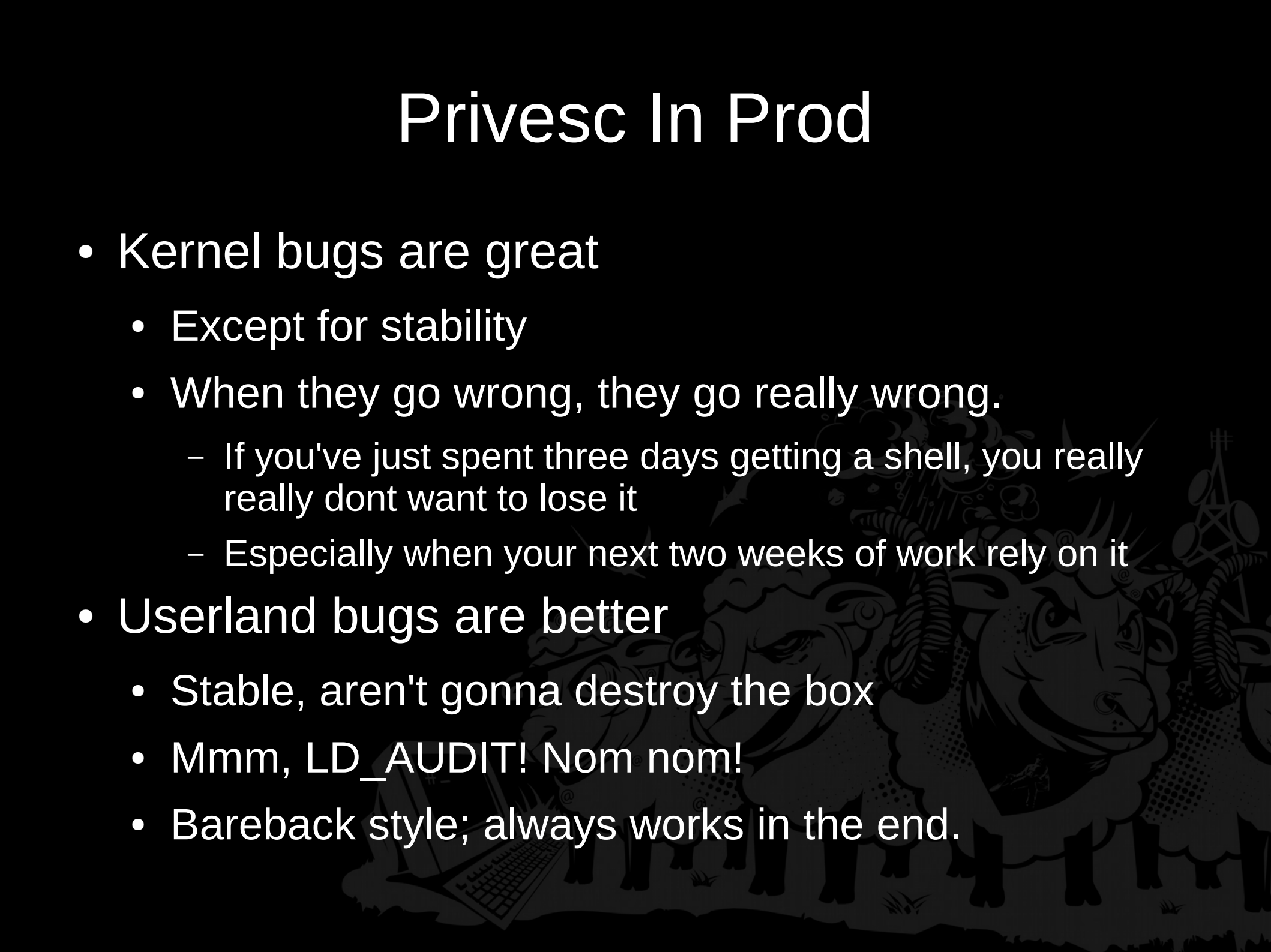
Restrictions & Realities

- But you can't really go in like a black hat; you have to care some.
  - make best effort to not break production
  - If you take out someone's prod box, chances are while it's a good lesson for everyone, you aint gonna get another round next FY...
- So when you want to try the latest kernel privesc from hawkes...



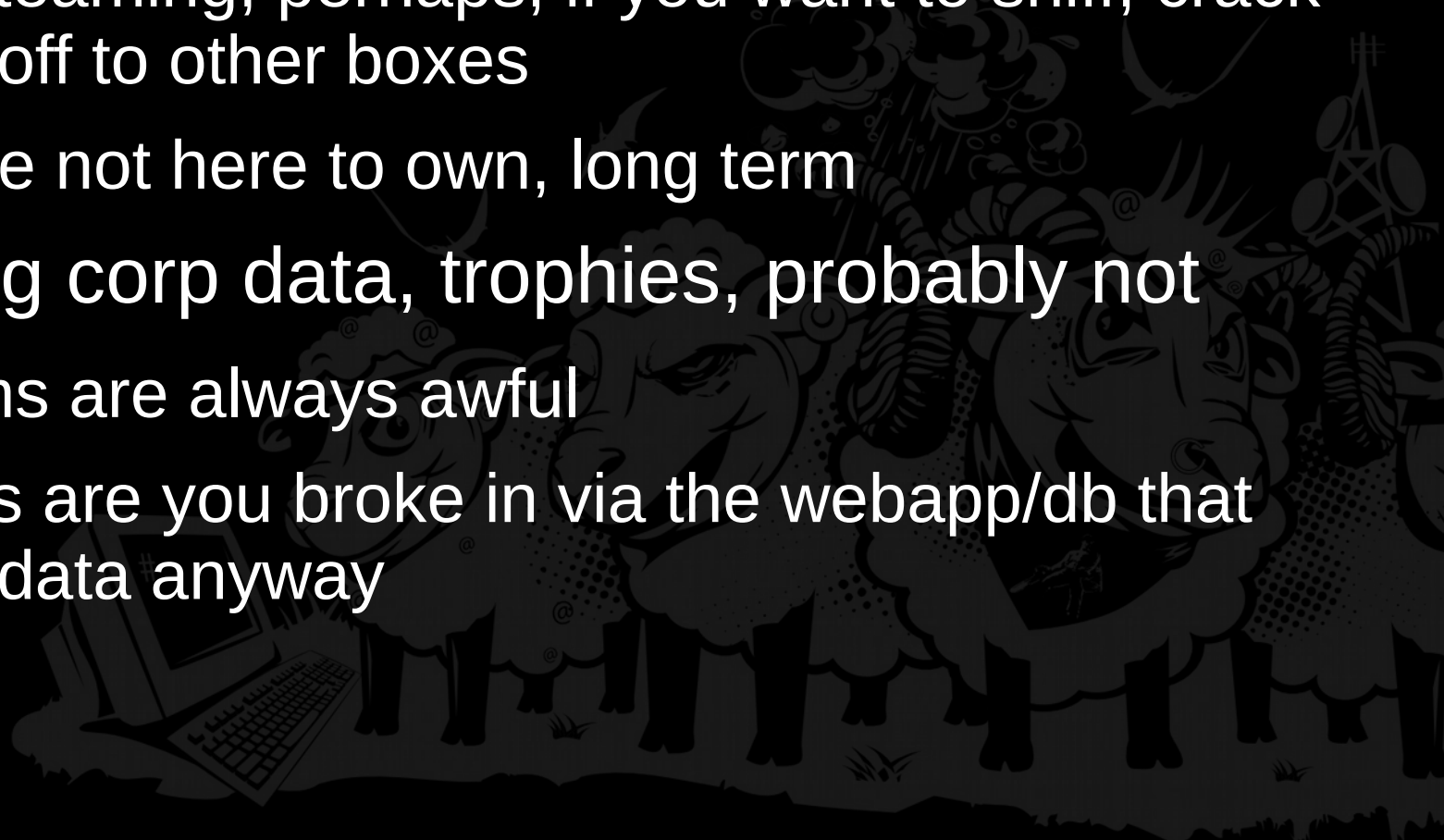
Privesc In Prod

- Kernel bugs are great
  - Except for stability
  - When they go wrong, they go really wrong.
    - If you've just spent three days getting a shell, you really really dont want to lose it
    - Especially when your next two weeks of work rely on it
- Userland bugs are better
  - Stable, aren't gonna destroy the box
  - Mmm, LD_AUDIT! Nom nom!
  - Bareback style; always works in the end.



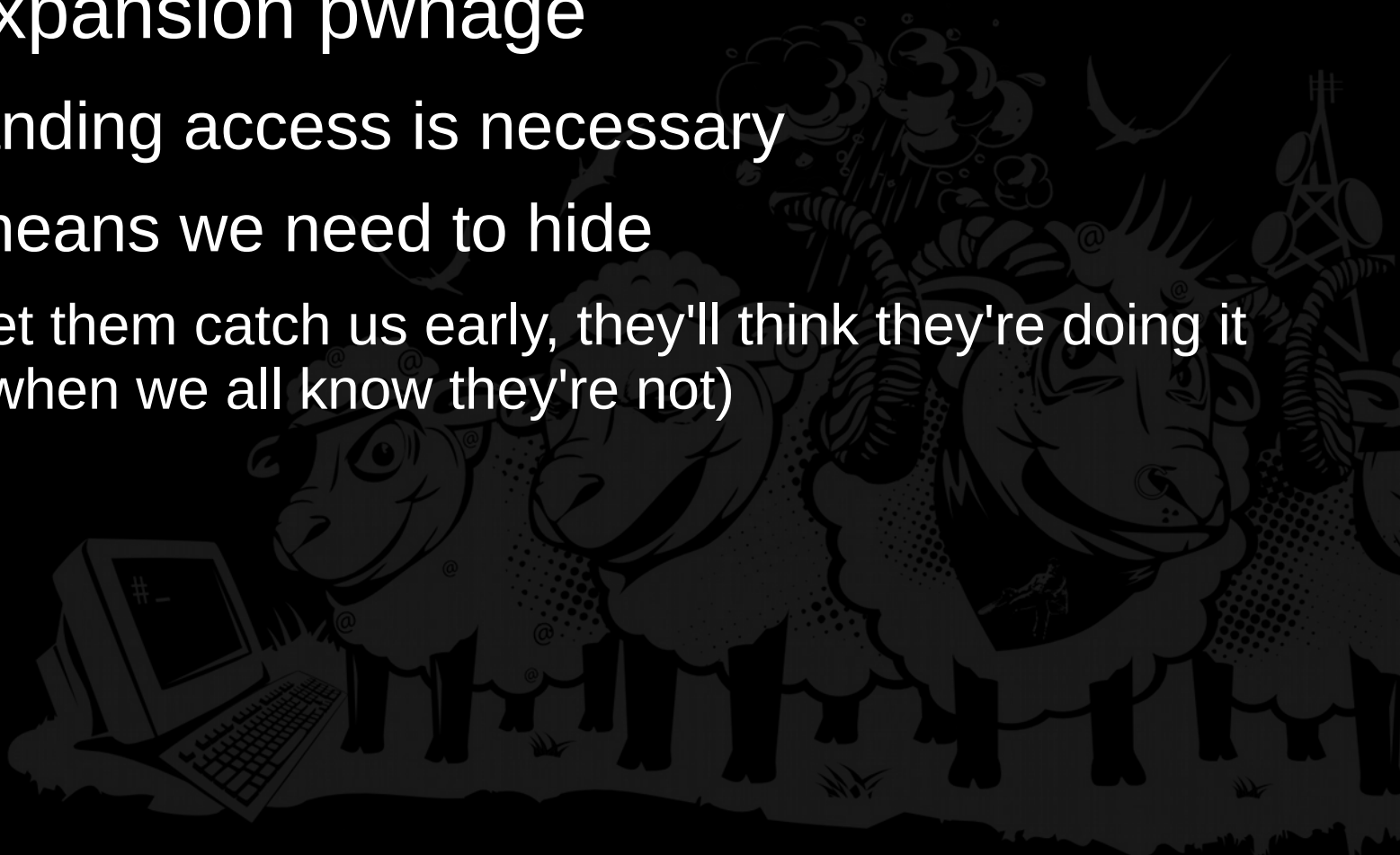
But, is Root a Distraction?

- Do you really need root?
  - For long term consilidation, prolly, yeah.
  - For red teaming, perhaps, if you want to sniff, crack or pivot off to other boxes
  - But we're not here to own, long term
- For getting corp data, trophies, probably not
  - FS perms are always awful
  - Chances are you broke in via the webapp/db that has the data anyway



Rootkits

- Are one good reason for root
- Red Teaming tests effectiveness of realworld internal expansion pwnage
  - So expanding access is necessary
  - Which means we need to hide
    - If we let them catch us early, they'll think they're doing it right (when we all know they're not)



Rootkit Features

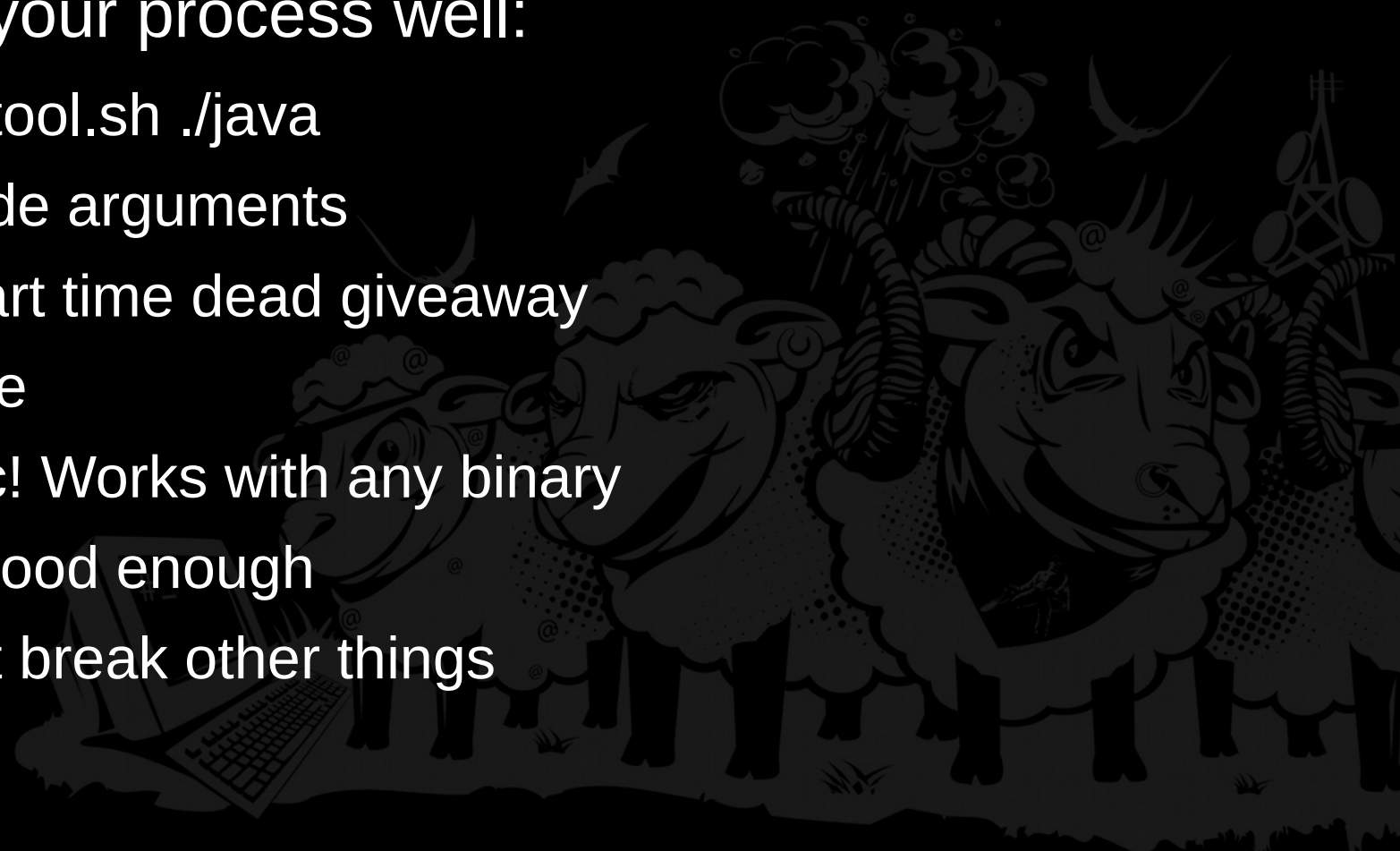
- In rough order of priority
  - Persistent access
  - Process hiding
  - File hiding
  - Socket hiding
- Last three all need root
  - Or do they?

Non-Rootkits

- Can we hide as non-root?
  - Sockets: not easily, no. Best we can do is have innocuous DNS PTR
  - Process … yes.
  - Files … yes.
- :)

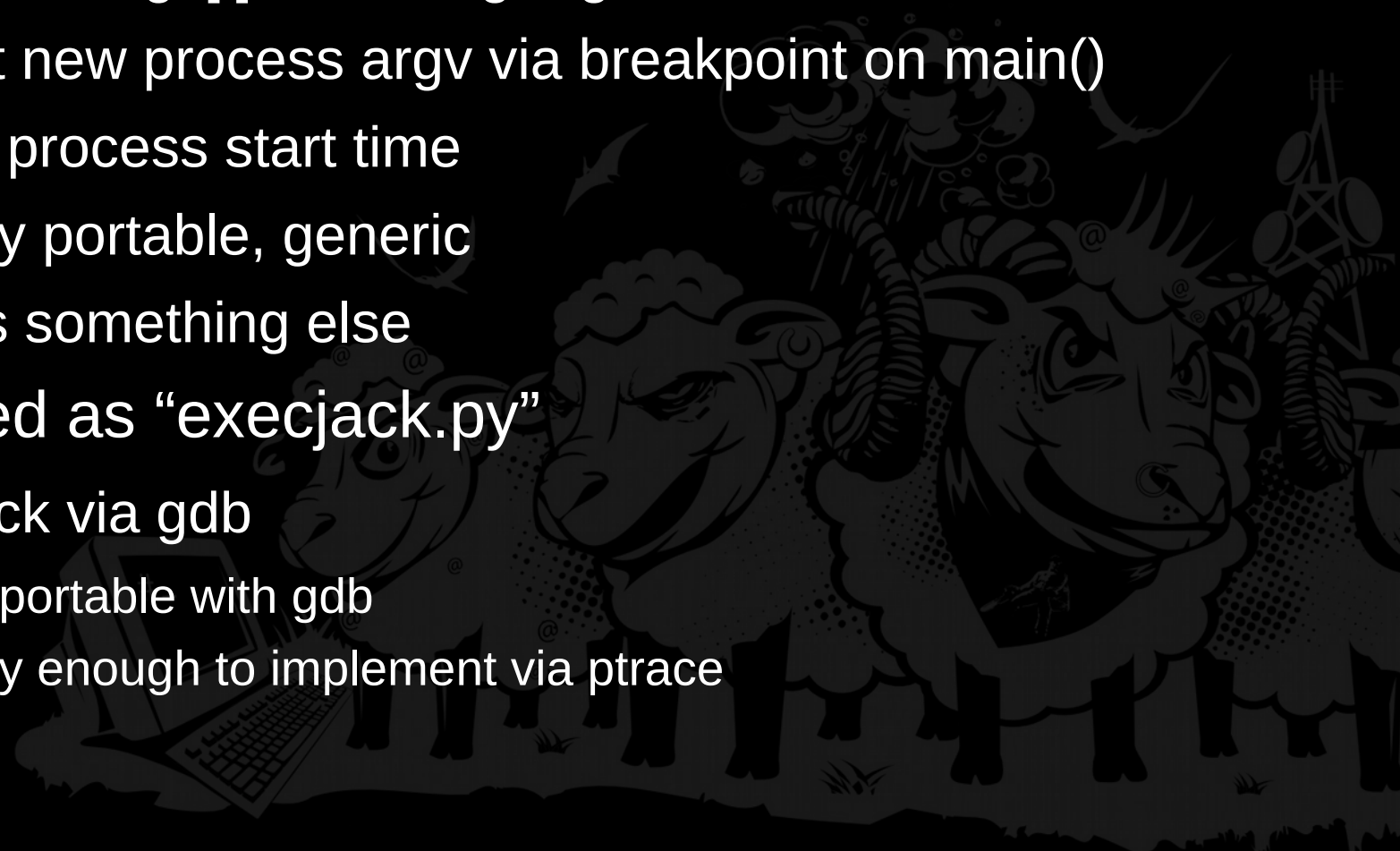
The Shell Game: Process

- We can't completely hide from ps list
- But we can hide in plain sight:
- L1: Name your process well:
  - Cp hax0rtool.sh ./java
  - -: can't hide arguments
  - -: proc start time dead giveaway
  - +: portable
  - +: generic! Works with any binary
  - +: easy, good enough
  - +: doesn't break other things



Level 2

- Ptrace() attach to process, inplace replace with exec() + argv stealth
  - Keep original argv[ ], including args
  - Switch out new process argv via breakpoint on main()
  - +: inherits process start time
  - +: still fairly portable, generic
  - -: replaces something else
- Implemented as “execjack.py”
  - Ala ssh-jack via gdb
    - no asm; portable with gdb
    - Also easy enough to implement via ptrace

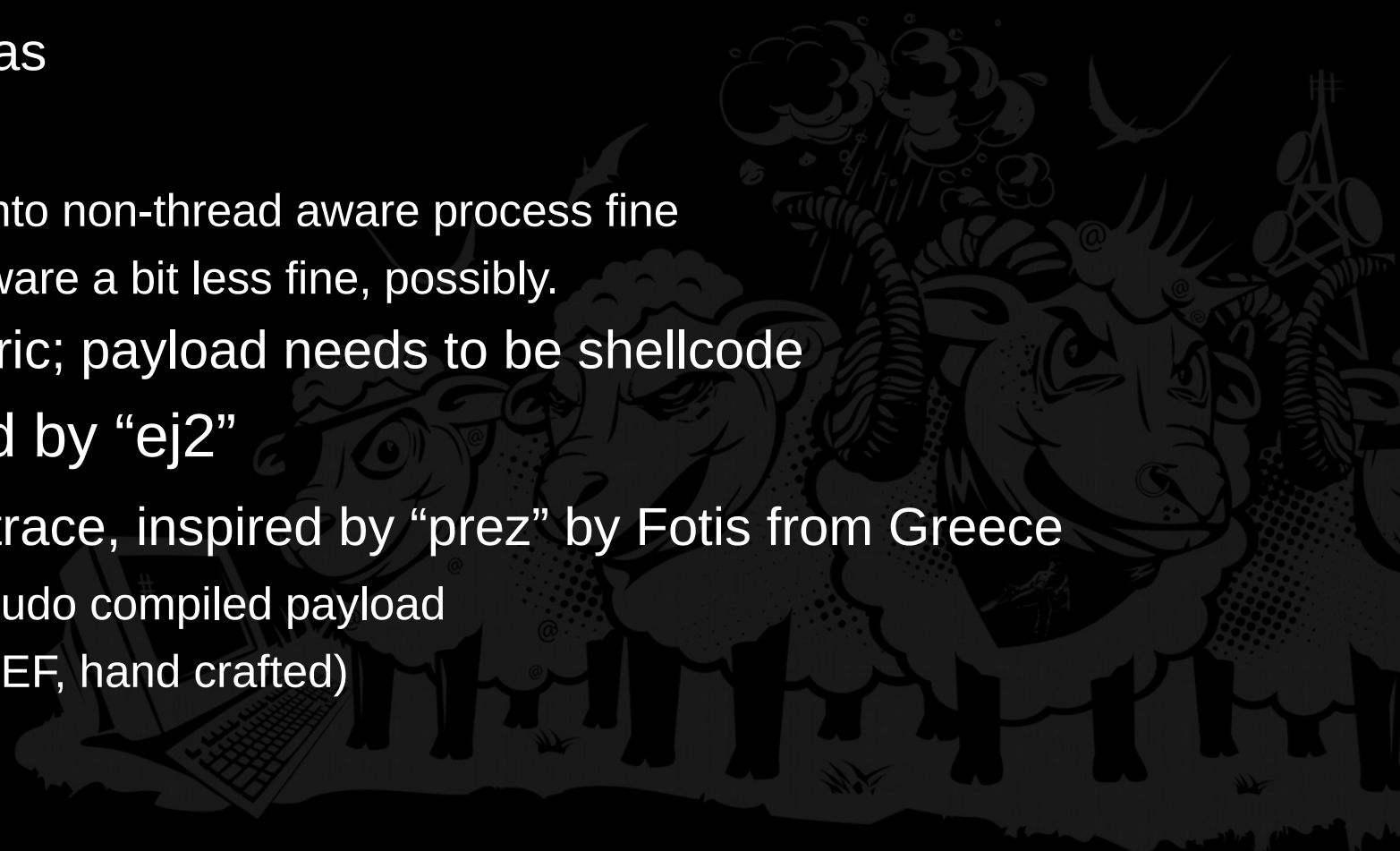


Demo

- Process hiding, Level 2:
  - Execjack vs ps

Level 3

- Ptrace() attach, clone(), inject code.
  - Starts payload in a new thread in host binary
  - Invisible to normal ps (without threads switch; eg ps -eLF)
  - +: stealthy as
  - -: Brittle
    - injection into non-thread aware process fine
    - Thread-aware a bit less fine, possibly.
  - -: Not generic; payload needs to be shellcode
- Implemented by “ej2”
  - Python + ptrace, inspired by “prez” by Fotis from Greece
    - Injects peludo compiled payload
    - (or MOSDEF, hand crafted)

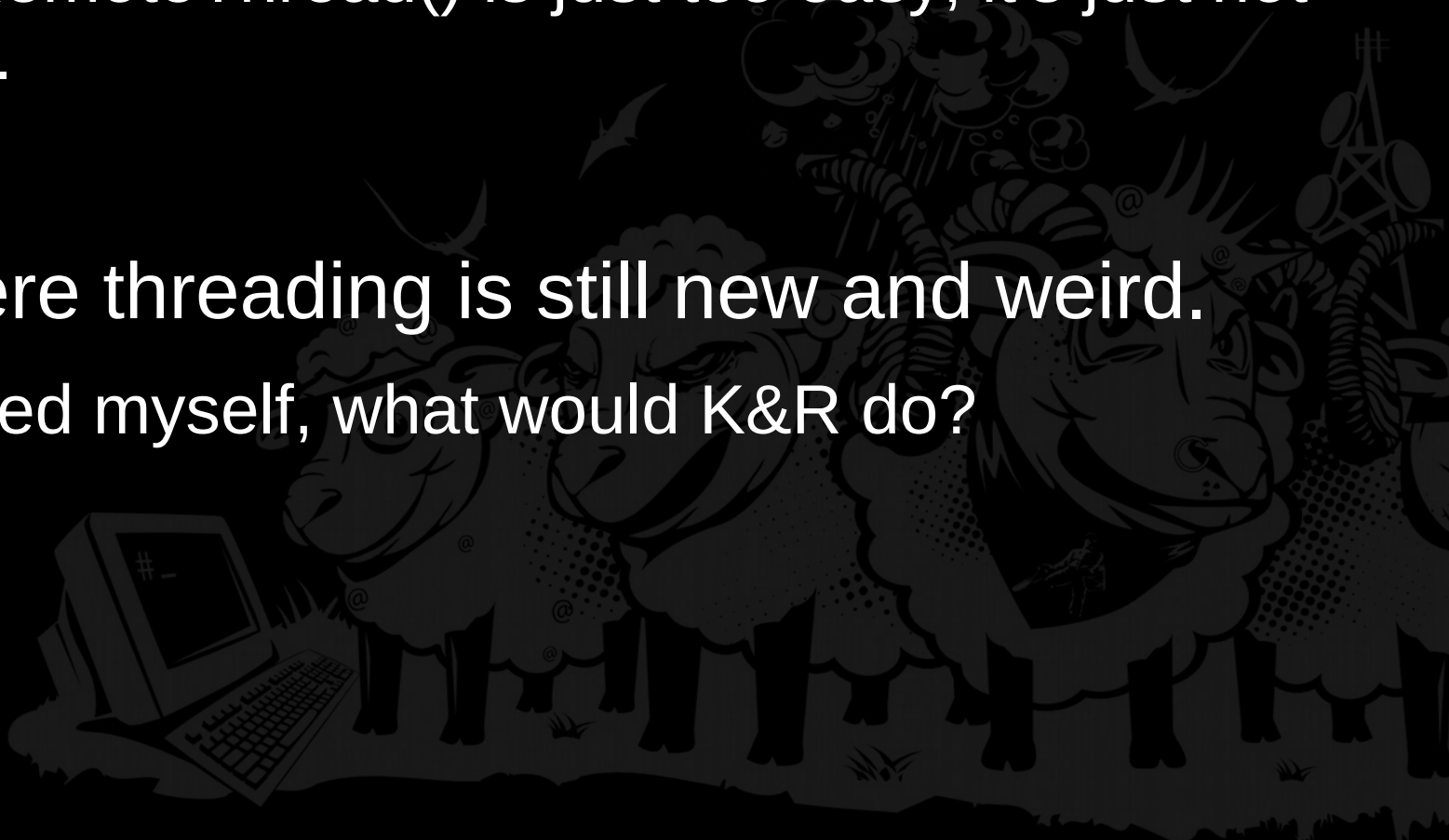


Demo

- Process hiding, Level 3:
  - ej2 vs ps

Hang On, Wait, err, Timeline?

- Welcome to like, Win32 circa 1995.
  - Yeah. Embaressing huh?
  - CreateRemoteThread() is just too easy; it's just not sporting.
- Unix: where threading is still new and weird.
  - So I asked myself, what would K&R do?

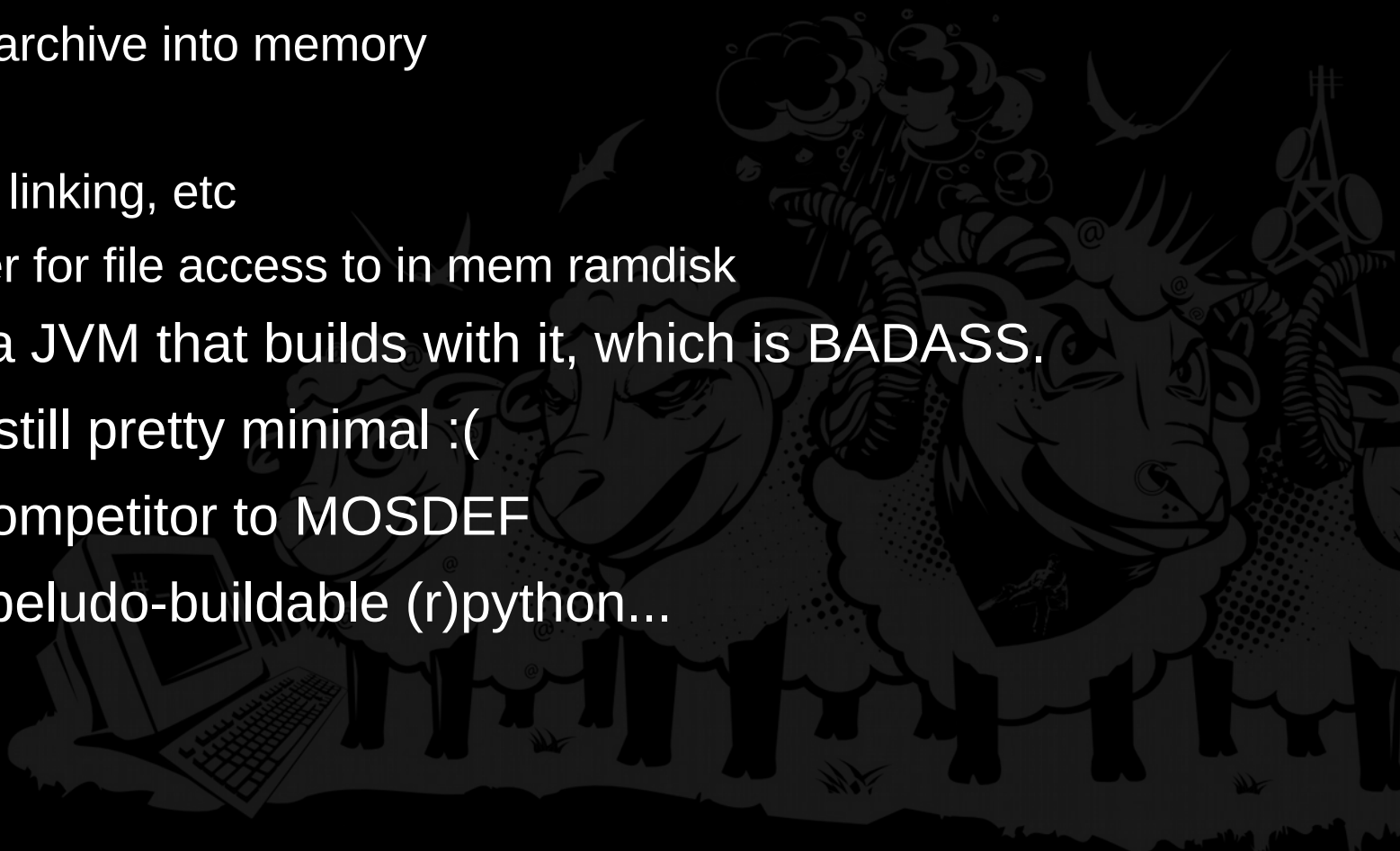


Level 4

- Ptrace() attach, inject parasitic cooperative backdoor
  - Get control periodically via:
    - signals (-ALRM, -VTALRM, -PROF) to get control
    - Hooking some key function (eg. Select)
  - Like a DOS TSR :)
  - +: really really not visible to ps, even less so than threaded
  - -: Fiddly; needs to assess best method for target binary
  - -: Payload needs to be shellcode
  - +: Doesn't impair normal operation (if we do it right)
- Not implemented yet :(
  - Ran outta time. It's half done. Sorry. Stupid Tokemon.

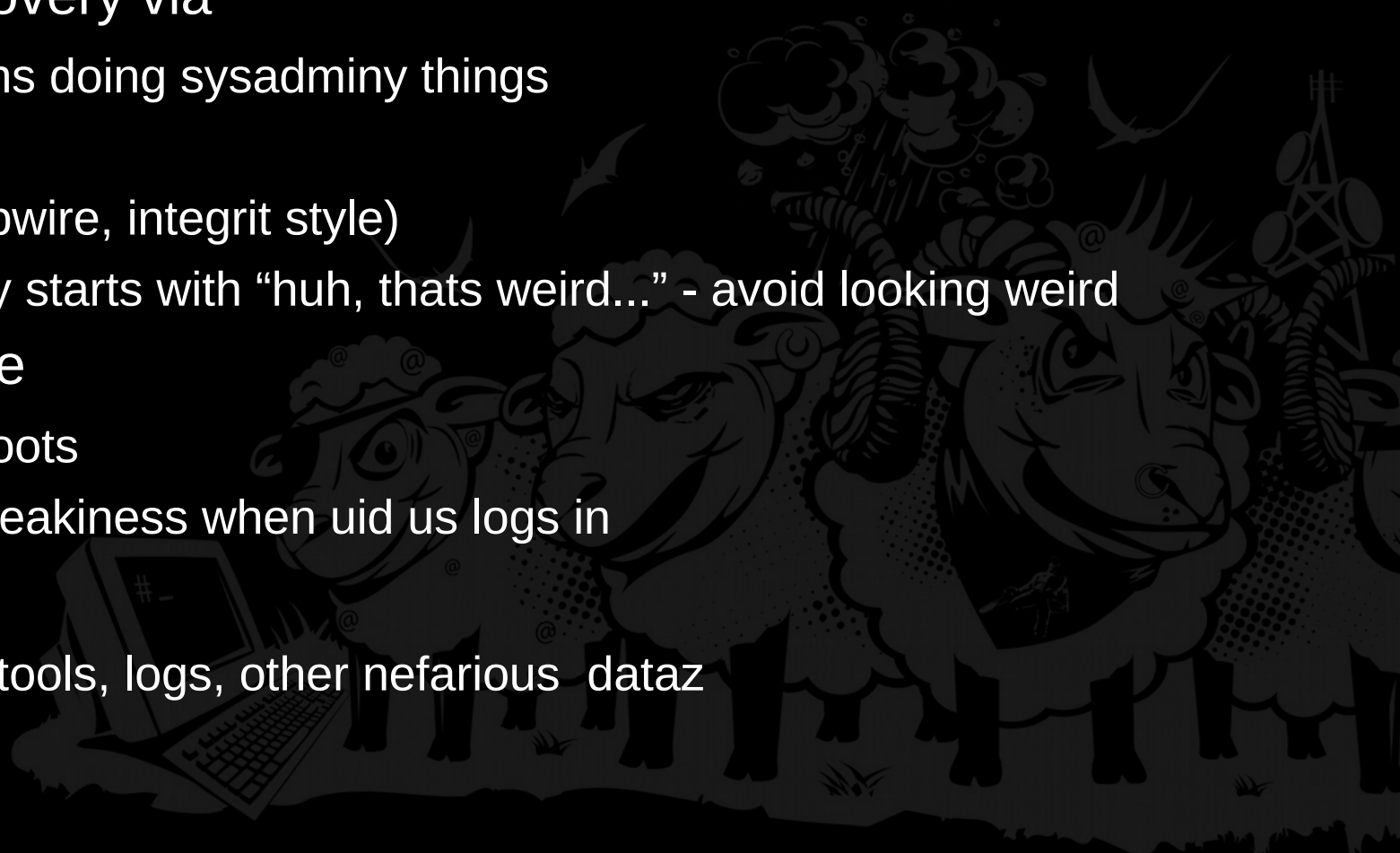
A Side Note: Peludo

- Part of the Netifera project
- GCC-based compiler toolchain for buliding self contained binaries
- Position independent
  - Shovel “.pld” archive into memory
  - Jmp to start
  - Takes care of linking, etc
  - Has VFS layer for file access to in mem ramdisk
- +: They have a JVM that builds with it, which is BADASS.
- -: Their libc is still pretty minimal :(
- Reasonable competitor to MOSDEF
- Now I want a peludo-buildable (r)python...



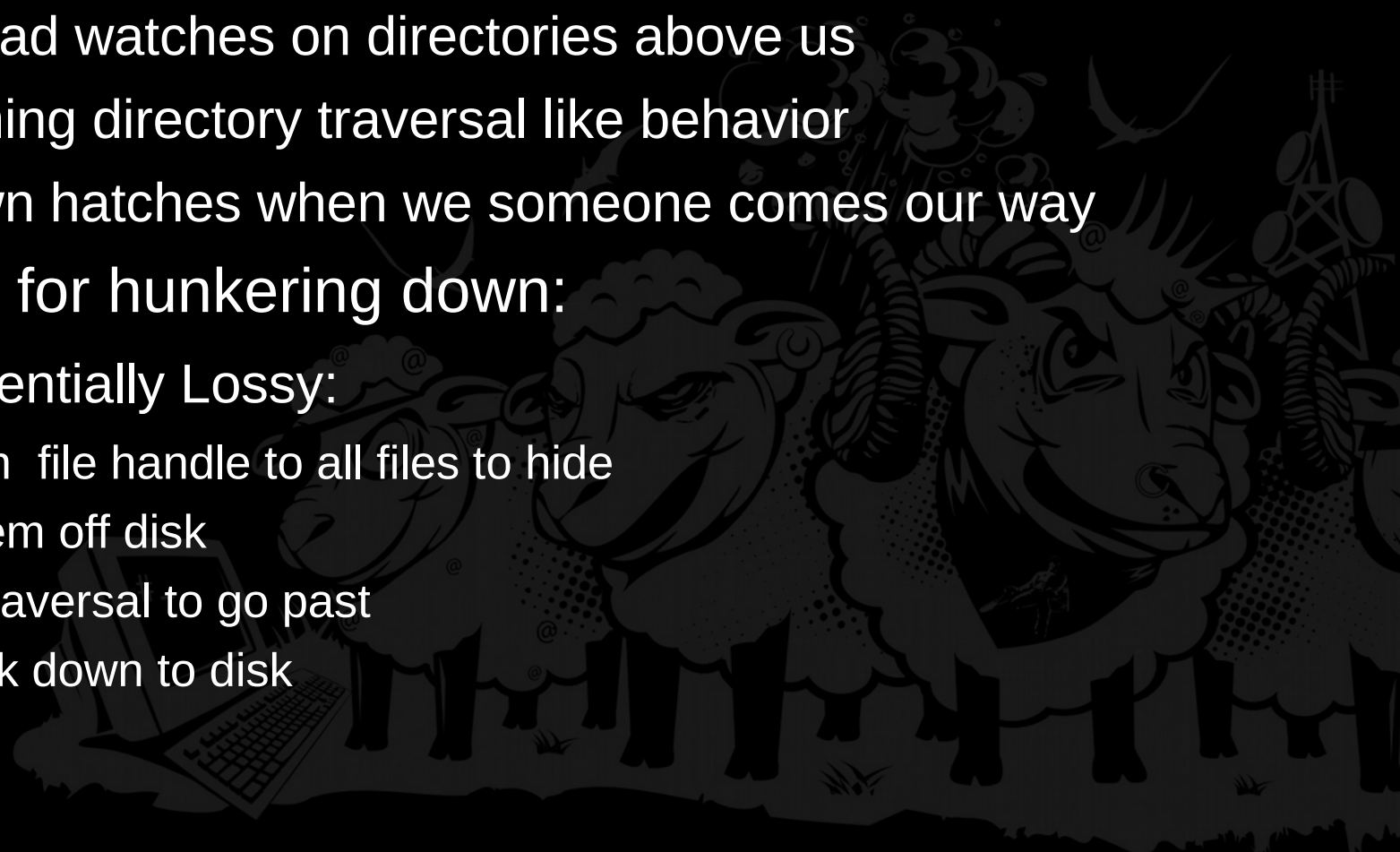
The Shell Game: Files

- How do we hide files from root, as non root?
- And why?
  - Avoid discovery via
    - Sysadmins doing sysadminy things
    - Backups
    - HIDS (tripwire, integrit style)
    - IR usually starts with “huh, thats weird...” - avoid looking weird
  - Persistence
    - Over reboots
    - To run sneakiness when uid us logs in
  - Stash
    - Hide our tools, logs, other nefarious dataz



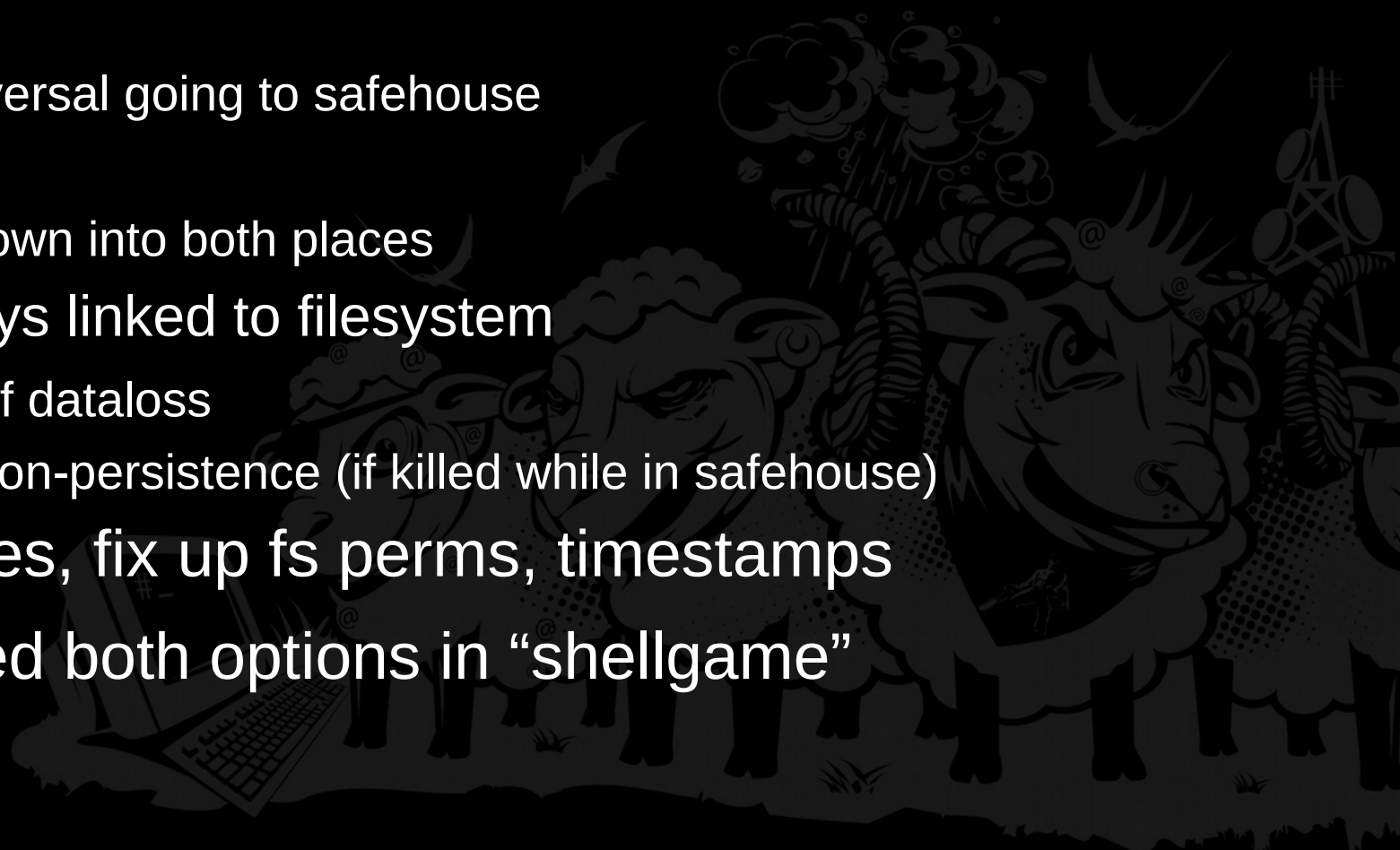
The Eponymous Shell Game

- Inotify based filesystem racer
  - Inotify is linux kernel infrastructure for registering fs-change callbacks
  - Register read watches on directories above us
  - Spot incoming directory traversal like behavior
  - Batten down hatches when we someone comes our way
- Two options for hunkering down:
  - Easier, Potentially Lossy:
    - Hold open file handle to all files to hide
    - Unlink them off disk
    - Wait for traversal to go past
    - Write back down to disk



Cont.

- Harder, Lossless:
  - Hardlink all files to other end of disk (“safehouse”)
  - Unlink out of original place
  - Wait
  - Spot traversal going to safehouse
  - Switch
  - Relink down into both places
- Files always linked to filesystem
  - No risk of dataloss
  - Risk of non-persistence (if killed while in safehouse)
- In both cases, fix up fs perms, timestamps
- Implemented both options in “shellgame”



Does it Actually Work?

- Yup
  - Against tar, find, ls -R, updatedb, prolly even integrit
    - i.e. all the things a thorough sysadmin would use
  - But direct access still works, so our trojans can write their logs, ha ha.
  - Some care about choosing good locations to hide
    - enough dirs deep
    - Safehouse far enough away
  - Could hammer disk to make it slower :)
  - Basically good enough (got a better idea?)
- It'll work against most syadmins, I'll wager...

DEMO

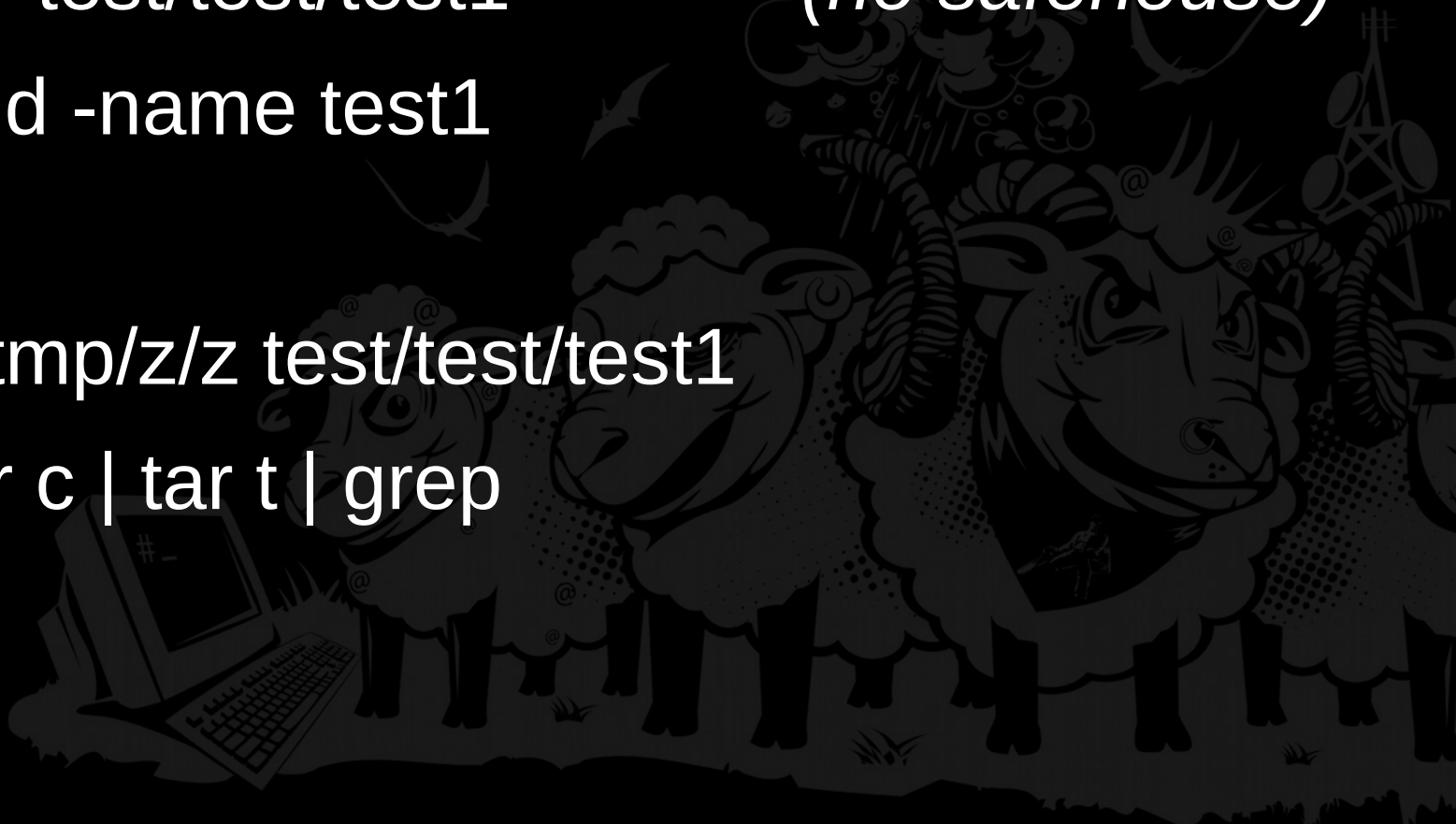
- Demotiem nao!

shellgame -r test/test/test1 *(no safehouse)*

vs. find -name test1

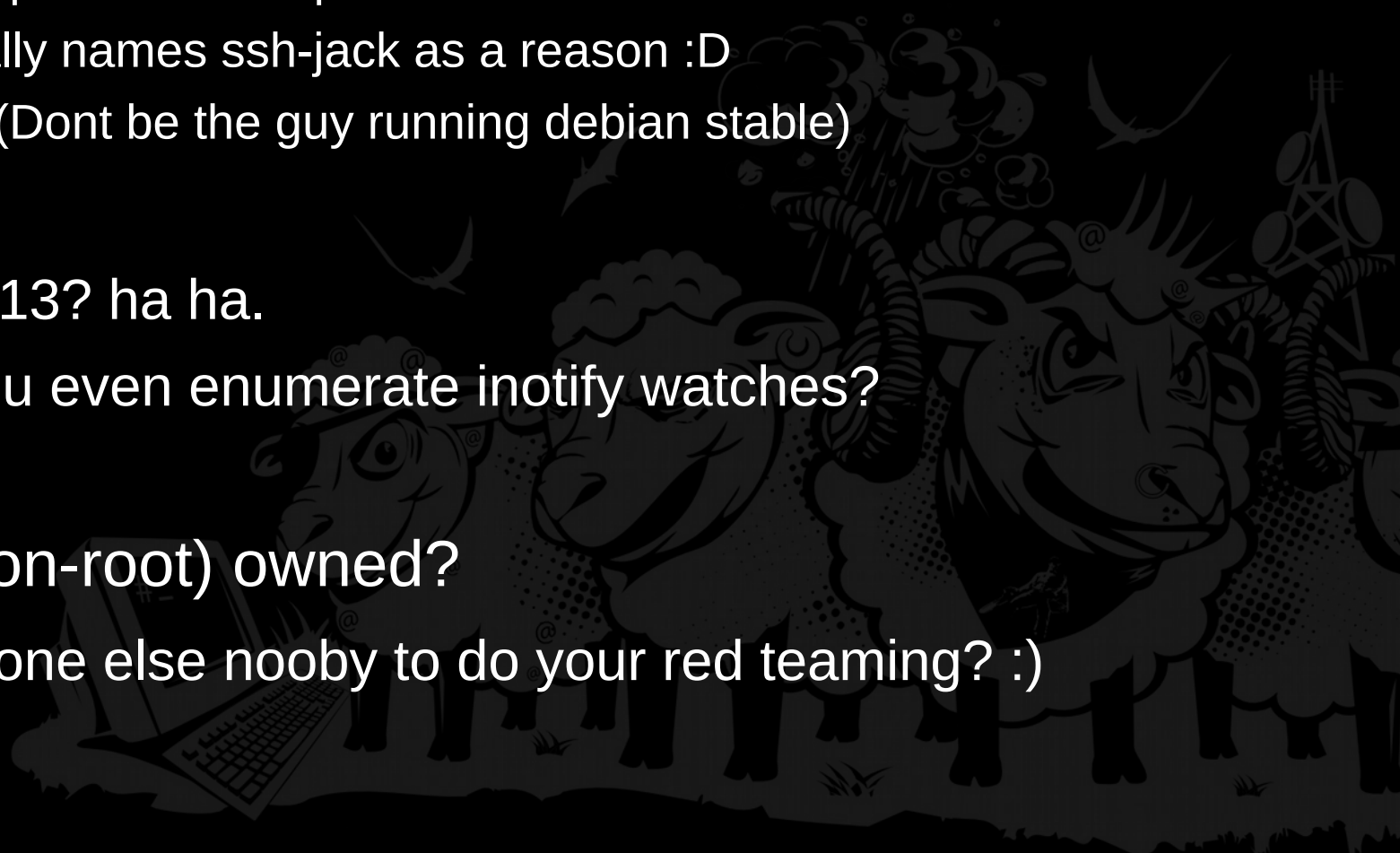
shellgame /tmp/z/z test/test/test1

vs. tar c | tar t | grep



Countermeasures

- Execjack
  - Modern ubuntu (>=10.10) has restricted ptrace kernel option
    - Can only ptrace a child process
    - Specifically names ssh-jack as a reason :D
    - Use this (Dont be the guy running debian stable)
- Inotify
  - Run < 2.6.13? ha ha.
  - How do you even enumerate inotify watches?
  - Dunno :)
- Dont get (non-root) owned?
  - Hire someone else nooby to do your red teaming? :)



That's That

- Kthx
- Code will be up on storm.net.nz somtime
  - After freakin' tokemon is gone.
  - Also see http://fotis.loukos.me/ for PreZ injector
- Questions
- I didn't let WiteRabit proof my slides, so I am the winnar

(Hope you're enjoying Kiwicon 4!)