# X86 linux kernel exploitation

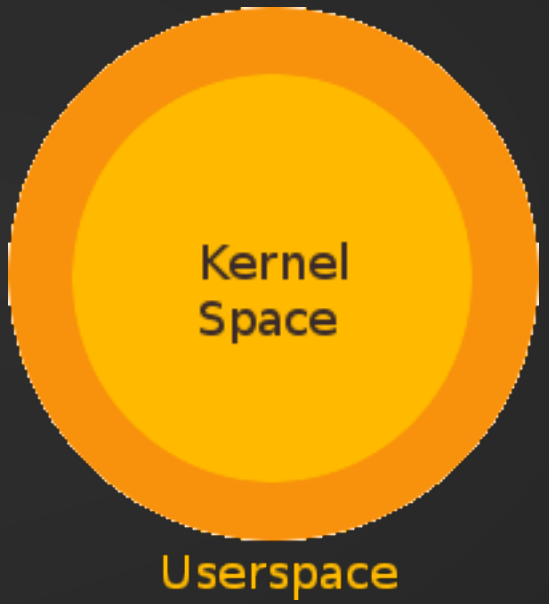
Guillaume Francqueville

ENSIBS – Thales Services

CTF : Team P_TE

# Kernelspace ? Userspace ?

- Vos applications tournent dans le userspace
- Userspace :
  - I/O avec le filesystem
  - Processus
  - Etc
- Possible grâce au kernel !
- Kernel:
  - contrôler et interagir avec le hardware
  - Fournir  un **environnement** au **applications**

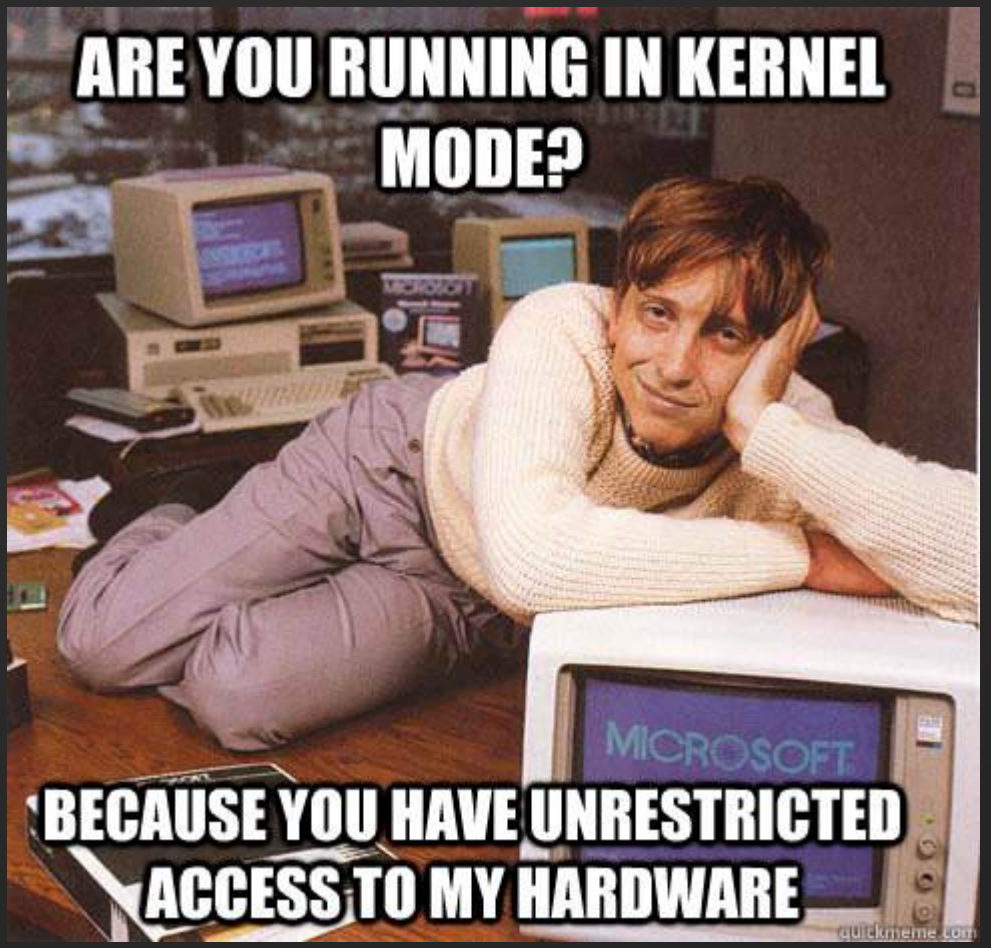


# Exploiter le kernel

- Les vulnérabilités du userspace sont aussi dans le kernel space
  - Buffer overflow
  - Use-after-free
  - Etc
- Modules kernel = excellents points d'entrées
  - LKM (Loadable Kernel Module) sont des binaires comme les classiques ELF, PE, MACH-O..
  - "drivers" pour filesystem, réseau, etc
  - Lsmod liste vos modules kernel
- But : rooter la machine !

# Stratégie

- Faire executer du code
  - Même méthode que pour les ELF,PE, etc
- Élever ses priviléges
- Retourner dans le userspace
- Enjoy your shell

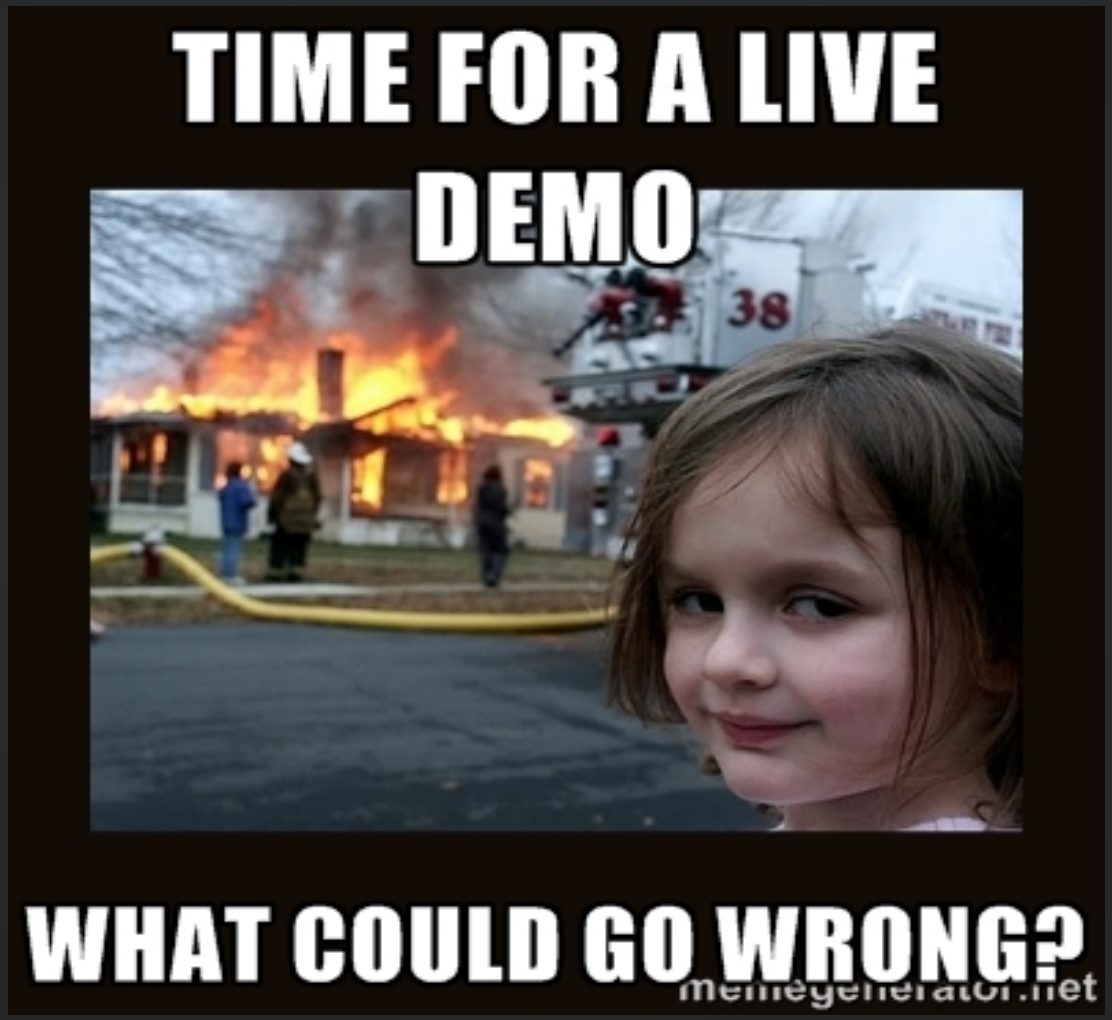


# Retour vers le userland

- On ne souhaite pas rester dans le kernelspace
  - Bug kernel = reboot
  - Les fonctionnalités usuelles sont plus faciles à utiliser dans le userspace
- Comment s'y prend le kernel ?
  - Instruction "iret"
  - Une stack en bonne et due forme
    - Push %ss (Stack Segment)
    - Push  @stack
    - Push %eflags
    - Push %cs (Code segment)
    - Push @fonction

# Démo time !

- Lancer la vm : ./run.sh (require qemu-system-x86)



# Et maintenant ?

# Les sécurités kernel

- Vous connaissez NX, ASLR, RELRO, PIE, CANARY ?
  => le kernel a lui aussi ses sytèmes de défense !
- mmap_min_addr
  - empêche les null pointer dereference
- Kallsyms
  - Empêche de trouver les adresses des fonctions prepare_kernel_cred et commit_creds
- Et d’autres !