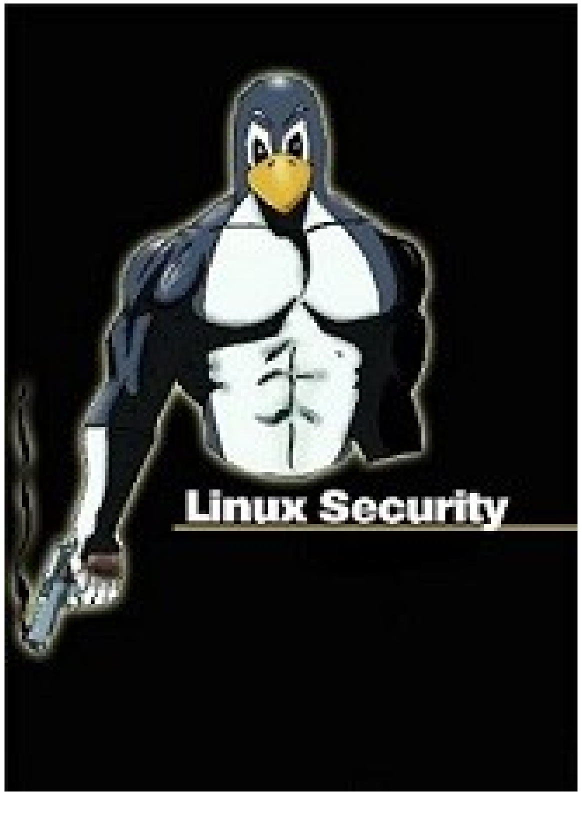
Linux Security



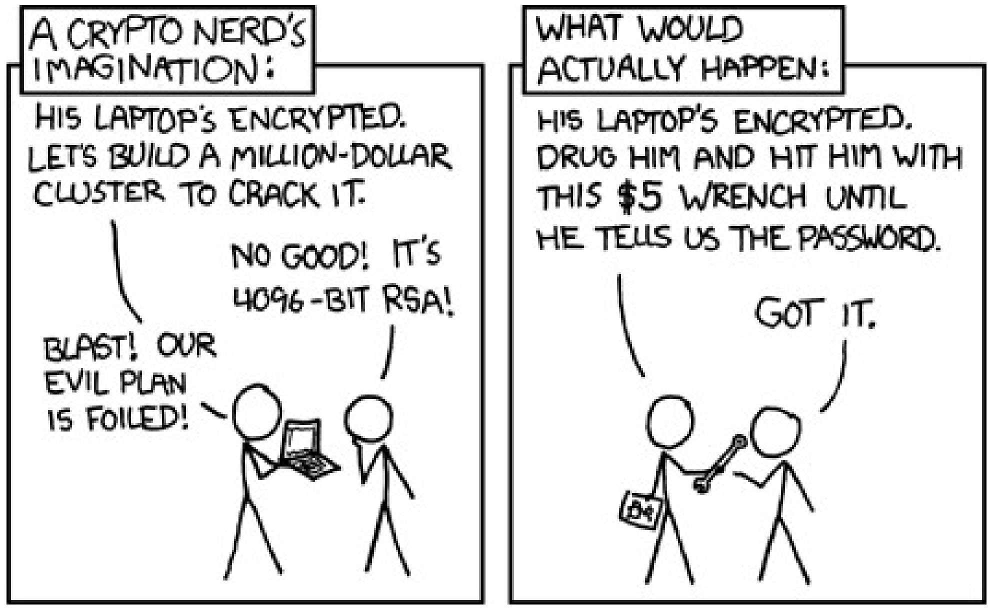
Scope

- General security practices
- Actionable steps
- Linux server & workstation security

Threat Model

- Three Main Considerations
  - Who are you defending against?
  - What capabilities do they have?
  - How persistent will they be?
- The Rubber Hose Test

- A system is secure under some threat model when the cost of breaking the security is greater than the attacker's resources
  - or motivation


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Threat Model

- NSA
  - Virtually unlimited motivation
  - Virtually unlimited resources
  - They probably backdoored whatever you’re using anyway
    - Dual EC, Intel IME, Android baseband, etc
  - You always lose

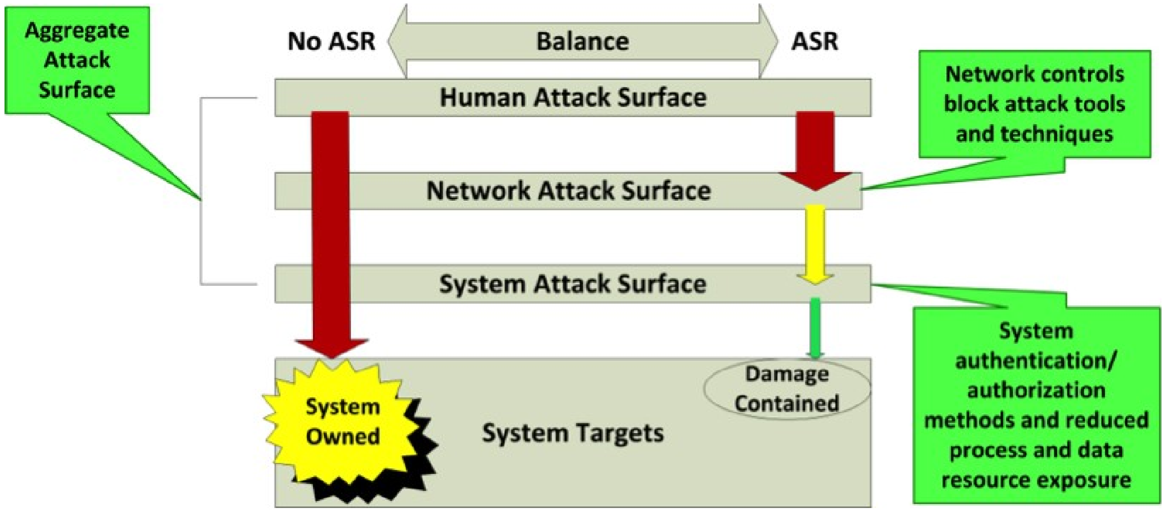
- TSA, customs
  - Limited motivation
  - Limited resources

Threat Model

- My point is that security people need to get their priorities straight. The "threat model" section of a security paper resembles the script for a telenovela that was written by a paranoid schizophrenic: there are elaborate narratives and grand conspiracy theories, and there are heroes and villains with fantastic (yet oddly constrained) powers that necessitate a grinding battle of emotional and technical attrition. In the real world, threat models are much simpler (see Figure 1). Basically, you're either dealing with Mossad or not-Mossad. If your adversary is not-Mossad, then you'll probably be fine if you pick a good password and don't respond to emails from ChEaPestPAiNPi11s@virus-basket.biz.ru. If your adversary is the Mossad, YOU'RE GONNA DIE AND THERE'S NOTHING THAT YOU CAN DO ABOUT IT. The Mossad is not intimidated by the fact that you employ https://. If the Mossad wants your data, they're going to use a drone to replace your cellphone with a piece of uranium that's shaped like a cellphone, and when you die of tumors filled with tumors, they're going to hold a press conference and say "It wasn't us" as they wear t-shirts that say "IT WAS DEFINITELY US," and then they're going to buy all of your stuff at your estate sale so that they can directly look at the photos of your vacation instead of reading your insipid emails about them. In summary, https:// and two dollars will get you a bus ticket to nowhere. Also, SANTA CLAUS ISN'T REAL. When it rains, it pours.
  - James Mickens

Vectors

- Attack vector
  - Entry point an attacker may use to probe or exploit
  - Generally any place where an attacker can influence the state of the system
  - Physical access
  - Mental access
  - I/O
- Attack surface
  - Sum of all attack vectors



Case Study: short.csc.ncsu.edu

- Without server room access
  - OpenSSH server
  - lug.ncsu.edu
  - Apache2
  - Virtualbox
  - ZNC
  - ?
- With access
  - USB
  - NIC
  - Power
  - ?

Linux Security Model

- Authentication
  - Is the user authorized to access the system?
- Permissions
  - Filesystem, user based
- Capabilities
  - Way to give certain binaries certain privileges
- Configuration

- Out of the box, Linux is fairly secure
- Many improvements can be made

Users & Groups

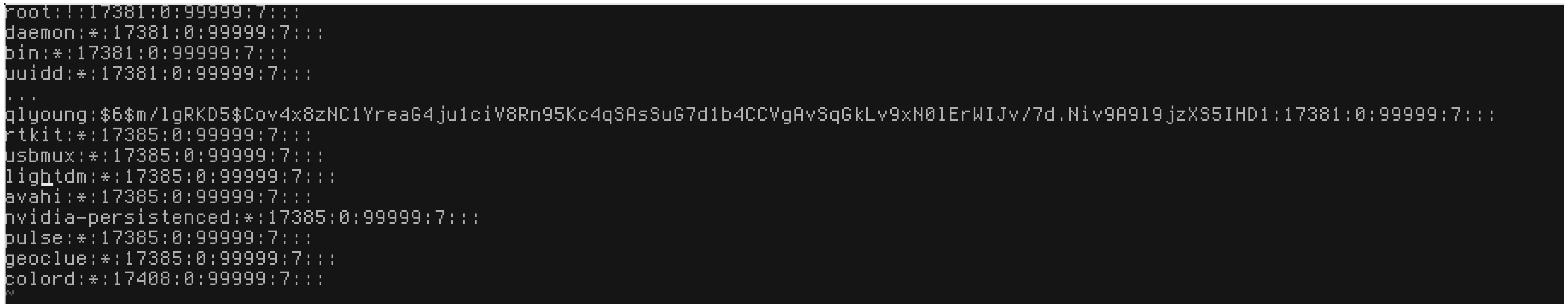
- Individual entity on system
  - Internally represented by UIDs = user IDs
- Has set of permissions
- Can be set as ‘owner’ of files & set permissions on owned files
- Groups = way of granting permissions, policy to a set of users
  - Internally represented by GIDs = group IDs

- ‘root’ / ‘superuser’
  - Similar to Administrator or SYSTEM on Windows
  - UID = 0
- Fundamental unit in security

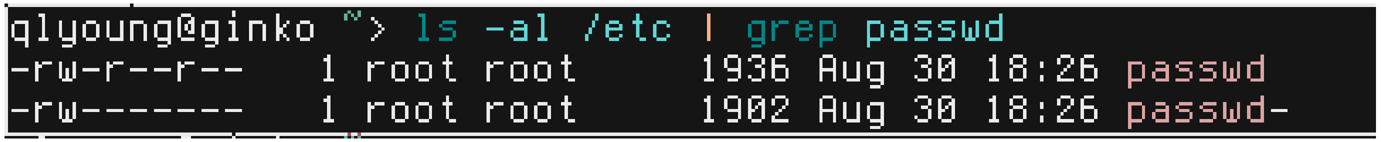
Authentication

- Two types: remote & local
- Generally only concerned with remote auth
  - Local implies physical access
- Various ways of implementing
  - OOTB: /etc/[passwd,shadow]
  - Pluggable Authentication Modules (PAM)
  - LDAP, SSSD, Kerberos...
- NCSU uses OpenLDAP

- Linux stores hashed passwords in /etc/shadow
  - Typically world-readable, root-owned
  - user:$algo$salt$hash:…
  - * or ! or !! = disabled / locked
  - Ex. root is disabled



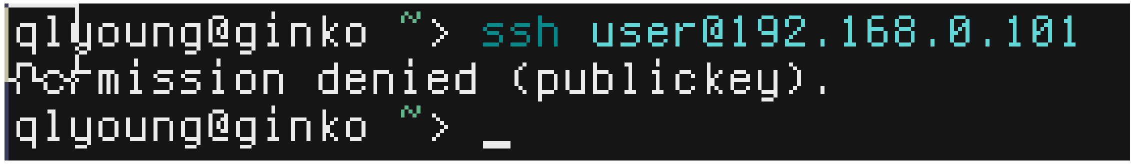
```
root:!:17381:0:99999:7:::
daemon:*:17381:0:99999:7:::
bin:*:17381:0:99999:7:::
uuidd:*:17381:0:99999:7:::
...
qlyoung:$6$m/lgRKD5$Cov4x8zNC1YreaG4ju1ciV8Rn95Kc4qSAsSuG7d1b4CCVgAvSqGkLv9xN01ErWIJv/7d.Niv9A919jzXS5IHD1:17381:0:99999:7:::
rtkit:*:17385:0:99999:7:::
usbmux:*:17385:0:99999:7:::
lightdm:*:17385:0:99999:7:::
avahi:*:17385:0:99999:7:::
nvidia-persistenced:*:17385:0:99999:7:::
pulse:*:17385:0:99999:7:::
geoclue:*:17385:0:99999:7:::
colord:*:17408:0:99999:7:::
```



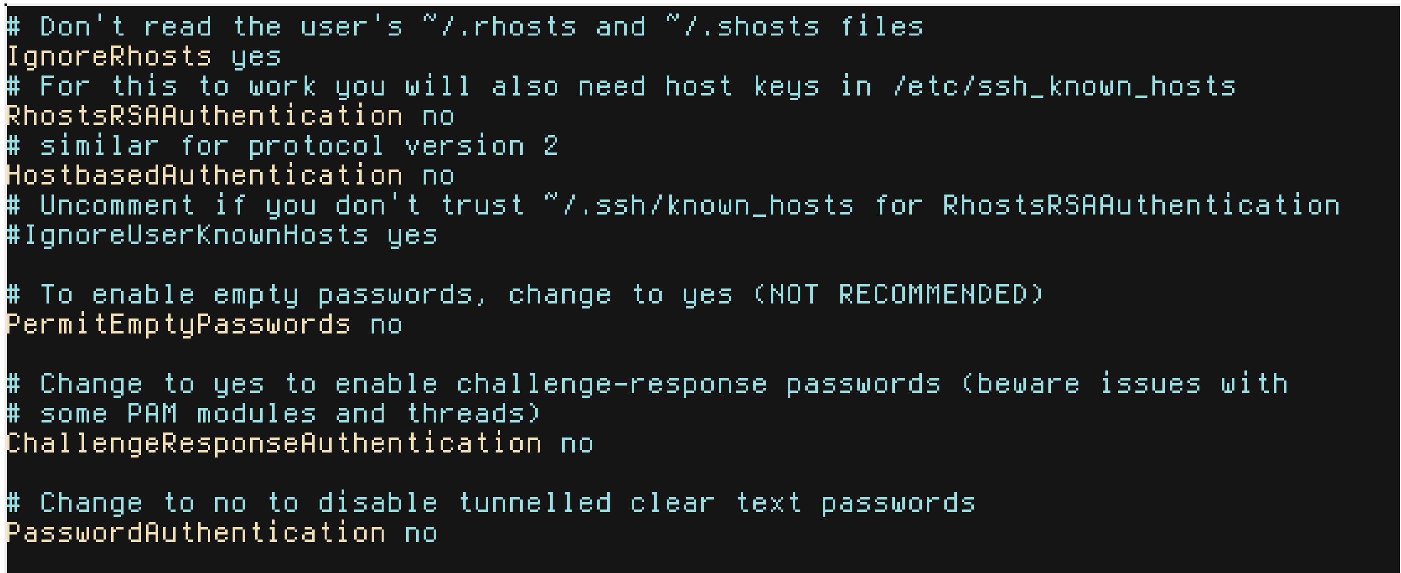
```
qlyoung@ginko ~> ls -al /etc | grep passwd
-rw-r--r--    1 root root      1936 Aug 30 18:26 passwd
-rw-------    1 root root      1902 Aug 30 18:26 passwd-
```

Authentication

- Remote: SSH
  - Same methods of backend auth
  - Many ssh-unique factors
  - Keypair auth
  - Port knocking
  - IP blocking
- Passwords
  - Complexity
  - Cycling



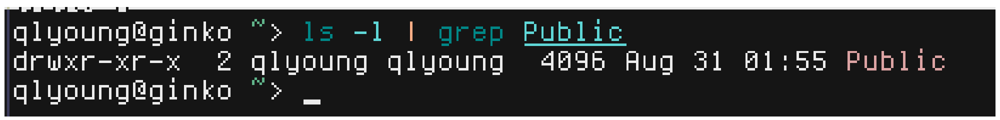
```
glyoung@ginko ~> ssh user@192.168.0.101
Permission denied (publickey).
glyoung@ginko ~> _
```



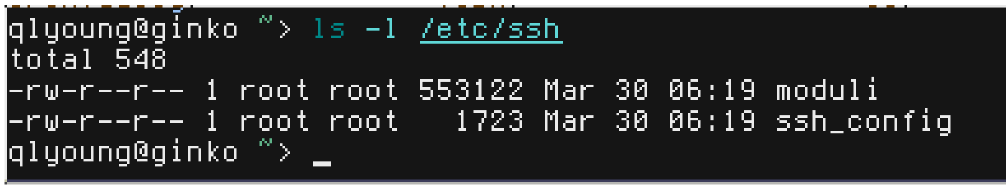
```
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no
```

Permissions

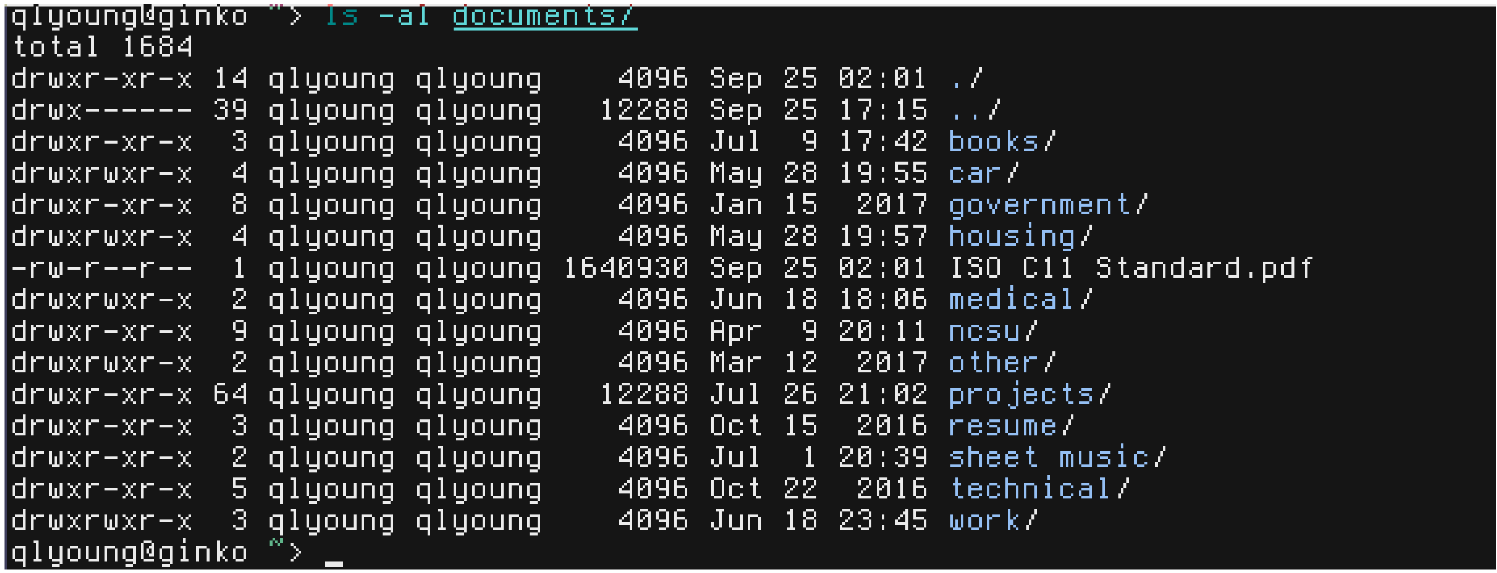
- Linux's way of determining which users can access which files and how
  - Read / Write / Execute
- System files owned by root



```
qlyoung@ginko ~> ls -l | grep Public
drwxr-xr-x  2 qlyoung qlyoung  4096 Aug 31 01:55 Public
qlyoung@ginko ~> _
```



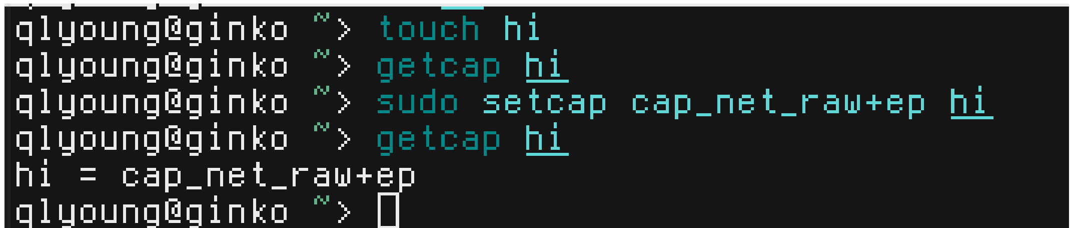
```
qlyoung@ginko ~> ls -l /etc/ssh
total 548
-rw-r--r-- 1 root root 553122 Mar 30 06:19 moduli
-rw-r--r-- 1 root root   1723 Mar 30 06:19 ssh_config
qlyoung@ginko ~> _
```



```
qlyoung@ginko ~> ls -al documents/
total 1684
drwxr-xr-x 14 qlyoung qlyoung    4096 Sep 25 02:01 ./
drwx------ 39 qlyoung qlyoung   12288 Sep 25 17:15 ../
drwxr-xr-x  3 qlyoung qlyoung    4096 Jul  9 17:42 books/
drwxrwxr-x  4 qlyoung qlyoung    4096 May 28 19:55 car/
drwxr-xr-x  8 qlyoung qlyoung    4096 Jan 15  2017 government/
drwxrwxr-x  4 qlyoung qlyoung    4096 May 28 19:57 housing/
-rw-r--r--  1 qlyoung qlyoung 1640930 Sep 25 02:01 ISO C11 Standard.pdf
drwxrwxr-x  2 qlyoung qlyoung    4096 Jun 18 18:06 medical/
drwxr-xr-x  9 qlyoung qlyoung    4096 Apr  9 20:11 ncsu/
drwxrwxr-x  2 qlyoung qlyoung    4096 Mar 12  2017 other/
drwxr-xr-x 64 qlyoung qlyoung   12288 Jul 26 21:02 projects/
drwxr-xr-x  3 qlyoung qlyoung    4096 Oct 15  2016 resume/
drwxr-xr-x  2 qlyoung qlyoung    4096 Jul  1 20:39 sheet music/
drwxr-xr-x  5 qlyoung qlyoung    4096 Oct 22  2016 technical/
drwxrwxr-x  3 qlyoung qlyoung    4096 Jun 18 23:45 work/
qlyoung@ginko ~> _
```

Capabilities

- Feature introduced in Linux 2.2
- Before:
  - root = god
  - others = not god
- After:
  - root = god
  - qlyoung = demigod
  - apache = web god
  - ssh = shell god
  - …
- Splits up root permissions into individual “capabilities”
- Programs can set and unset capabilities
  - CAP_NET_ADMIN = various network privileges like adding routes, create & modify interfaces, etc
  - CAP_SYS_BOOT = reboot & exec kernels
- Extremely powerful & fine grained
- Not widely used by end users (or developers...)
- setuid / setgid
  - Special permission set on files
  - Allows them to execute with privileges of file owner or group
  - Commonly used to perform root tasks at startup and then “drop privileges”
- Capabilities are generally better than setuid / setgid
- man 7 capabilities



```
qlyoung@ginko ~> touch hi
qlyoung@ginko ~> getcap hi
qlyoung@ginko ~> sudo setcap cap_net_raw+ep hi
qlyoung@ginko ~> getcap hi
hi = cap_net_raw+ep
qlyoung@ginko ~> 
```

Configuration & Application Security

- Permissions, capabilities, authentication are nice, but...
- One misconfigured application can own the whole system
- Extremely important to properly configure **everything**
  - Difficult
  - Time consuming
  - Extremely easy to do it wrong

- Ex: misconfigured ssh
  - Specify that one particular host can log in as root without a password
  - All other protections irrelevant
- Application security very important
  - Suppose ssh runs as root
  - Someone finds remote exploitable bug that lets them overwrite ssh's code
  - Remote user now has root access

This is the Achilles heel of computer security

Security Modules

- Extensions that aim to put a sock on the Achilles heel
- SELinux
  - Kernel module that aims to separate policy away from implementation
  - Similar to capabilities but much more extensive
  - Hated by system administrators
  - Very fine-grained, knows about users, network ports, hardware devices, etc
  - Made by our best friends, the NSA :-)
- AppArmor
  - Much closer to capabilities
  - Implements a policy framework like SELinux
  - Focuses solely on applications
  - Present on Ubuntu OOTB
- grsecurity
  - Kernel patches that add things like stack cookies, ASLR, other hardening features
    - Many of these have since been added to mainline (unpatched) Linux
  - Unpopular with Linus because it breaks a ton of stuff
  - Doesn’t use LSM
- LSM
  - Linux Security Modules
  - Essentially an API for building things like SELinux & AppArmor

"The thing is a joke, and they are clowns. When they started talking about people taking advantage of them, I stopped trying to be polite about their bullshit. Their patches are pure garbage."

– Linus on grsecurity

SELinux

Practical Measures, User POV

- Disk encryption
  - LUKS
  - BitDefender
  - TrueCrypt :-(
- Password Managers
  - 1Password
  - LastPass
  - KeePass
  - pass (my fav)
- Passwords
  - Pwgen
- Two-factor auth
- SSH key pairs

Encryption & Cryptography

- Practice of enciphering data to render it unreadable to external parties
- Two general kinds
  - Symmetric crypto
  - Asymmetric crypto

- Ciphertext
- Plaintext
- Key

Symmetric Cryptography

- Symmetric encryption algorithm
- plaintext + key → ciphertext
- ciphertext + key → plaintext

- Advantages
  - Simple
  - Good security assuming key and algorithm are strong
- Disadvantages
  - Key sharing is hard

Asymmetric Encryption

- A.K.A. public-key cryptography
- Basic idea:
  - Each party has two keys
  - Public key, private key
  - To send me a message, you encrypt it with my public key
  - To decrypt your message, I decrypt it with my private key

- Advantages
  - Key sharing is trivial
  - Signing
- Disadvantages
  - Difficult for novice users

PGP

- Pretty Good Privacy
  - Phil Zimmerman, 1991
  - Canonical implementation of public-key cryptography using (usually) ElGamal and RSA
- Web of Trust
- Fingerprints

- GPG
  - Gnu Privacy Guard
  - GNU implementation of OpenPGP
  - Manages your private key(s) and the public keys of yourself and others

Q&A