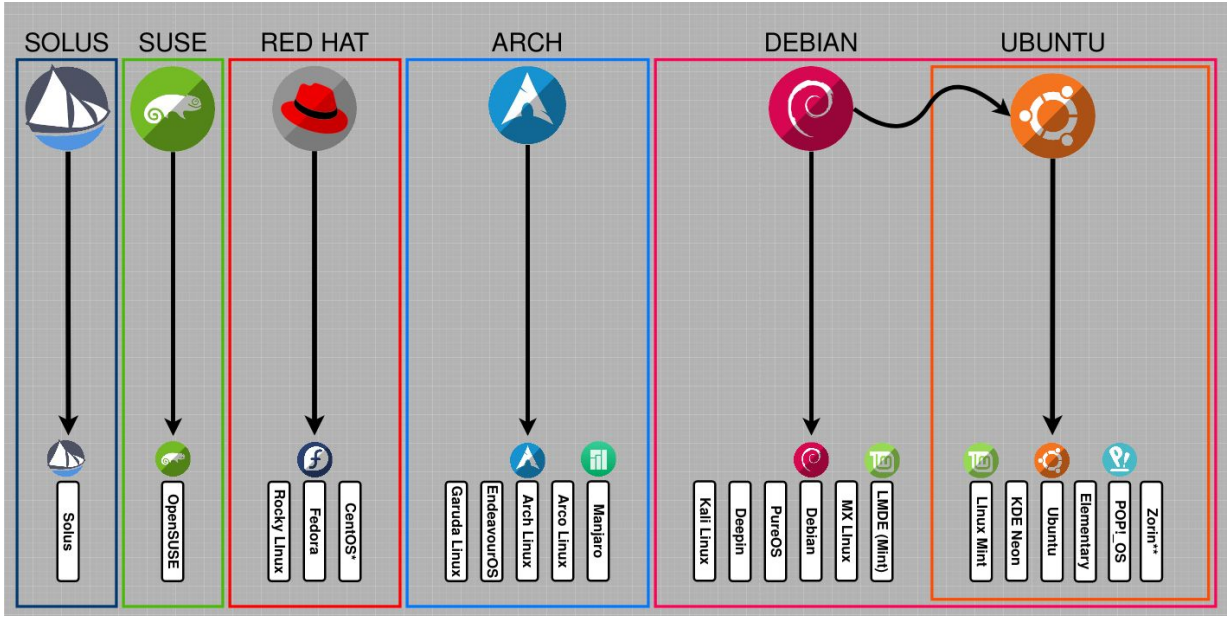
Linux

Server Exploits - Module 3

Overview

Linux

Linux is an open-source family of operating systems. They are primarily used for web servers, and servers that don't have much resources (CPU, RAM, etc). There are many flavours of Linux. We will primarily focus on Ubuntu, one of most popular distributions.



Linux

Linux is considered file-based. Where Windows is set up with many different objects, such as registry keys, policies, etc., Linux keeps all its settings in files. You can read these files, and write to them if you have the right permissions.

File System Overview

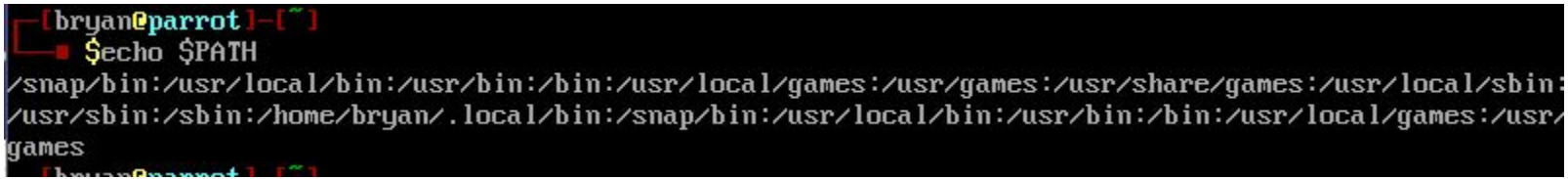
Linux File System

This is the typical file system in Linux. All dark-blue folders are major folders in the root file system. The root file system, or /, is the highest level of older structure.

Linux File System - /bin and /usr/bin

/bin is used where binaries, or executable files are sometimes kept depending on the distribution. In other distributions, such as Kali Linux, binaries are kept in /usr/bin, which is why on my machine /bin just is a pointer to /user /bin.

When you run a command on Linux, such as nmap, Linux looks in the current directory for that binary, then all other directories listed in your path. You can look at your path with the following command. Linux will look in /usr/bin, and other directories for the binary.



Linux File System - sudo

In the case of binaries, you may need root (essentially built-in administrator) permissions to run certain binaries. This is where the sudo (or **s**uper **u**ser **do**) command come in handy. Rather than login as the root user, one can execute a command with root permissions if they are part of the sudoers group. By default, the account made when you install Ubuntu is part of the sudoers group, but all new users will not be part when they are created.

You can become the sudo user with the sudo su command, however given that the root account has total access to all of the system, if you accidentally change a configuration it may be a pain to fix it. Be careful when running anything as the root user.

Linux File System - Popular Binaries

- systemctl - used to start, stop, etc services running on the machine. Must run with root permissions.
- apt - a package manager used on Ubuntu to install applications
- cd - change directory. Used to navigate throughout the Linux directory system.
  - cd .. will go back one directory.
  - cd on its own will go to the user's home directory
- ifconfig - deprecated in Linux. Use ip -a instead. ifconfig limits hardware addresses to 8 bytes, where newer devices bay use 20 bytes for addressing.
- man <command> - read the manual for any binary
- nano - good text editor
- pwd - display current directory's full path
- cp - copy file
- mv - Move file. Can also rename a file.
- mkdir - make a new directory
- cat - display contents of file
  - Good to use with grep, e.g. cat <filename> | grep <word>

Linux File System - Popular Binaries

- ls - will list contents of a directory.
  - ls -la - will list all contents of a directory (even hidden files) and display permissions

```
drwxr-xr-x. 4 root root    68 Jun 13 20:25 tuned
-rw-r--r--. 1 root root  4017 Feb 24  2022 vimrc
```

1. Directory or not, d or -
2. First 3 letters are for file owner. Can be red, write or execute.
3. Second 3 is applies to user group that owns file or directory
4. Third is all users on system
5. The number of directories inside a directory in the directory. Files will always be a 1.
6. User that owns the file or directory
7. Group that owns file or directory
8. Size of file or directory
9. When the file or directory was last modified

Linux File System - /boot

/boot contains files required for booting up your Linux system. These are really for the operating system and should never be touched.

Linux File System - /dev

/dev is used for device files. These are created when you plug in devices to your linux machine, such as a mouse or keyboard. The are used for controlling devices.

Linux File System - /etc

/etc used to be a dumping ground for miscellaneous files, hence the name, but now is mostly used for configuration files. If you install a program, like Apache web server, it's configuration files will be in here.

/etc also stores system configuration files, like your machine's hostname. Some important files are /etc/passwd (snippet shown below) and /etc/shadow. /etc/passwd shows all usernames, their IDs and their home directory and shell. My bryan account's home directory is /home/bryan, and my ID number is 1000 with my group's ID number being 1003. The root user (the built-in administrator user) has an ID of 0. The x is where a password hash would be on older systems. Nowadays, the users' hashes are stored on /etc/shadow, which only the root user can read.

Linux File System /home

/home is used for users’ (except the root user). This is where your Desktop, Downloads, etc. are stored.

Linux File System /lib

/lib is used to store libraries. When you are coding anything, and you import code from a library, this is where they are stored by default.

Linux File System /media

/media is where external storage, like an external hard drive, is mounted when you plug it in. you can use the cd command to navigate around it. Modern Linux distributions can automatically detect storage devices and mount them here.

Linux File System /mnt

/mnt is old and not really used anymore. In older versions of Linux, you would have to manually mount storage devices here.

Linux File System /opt

/opt is where software you compile, like source code off of Github, usually lands.

Linux File System /proc

/proc store information about your CPU, and other hardware. For example, /proc/meminfo stores information about your system’s memory.

Linux File System /run

/run is used by system processes to store temporary data. It is not recommended to touch this directory.

Linux File System /sbin and /usr/sbin

/sbin, or /usr/sbin stores binaries that can be run by the root (or super) user. This is why it starts with an “s”.

Linux File System /usr

/usr contains binaries, documentation, and anything else that needs to be shared by users on the system. For example, on your Kali machine, you can find wordlists under ‘/user/share. This directory is a good place for sharing files with other users that would be on your Kali machine.

Linux File System /srv

/srv is used for server files. This would be where you store web server files (HTML, JS, CSS files). If you had an FTP server, files uploaded would go to /srv/ftp.

Linux File System /sys

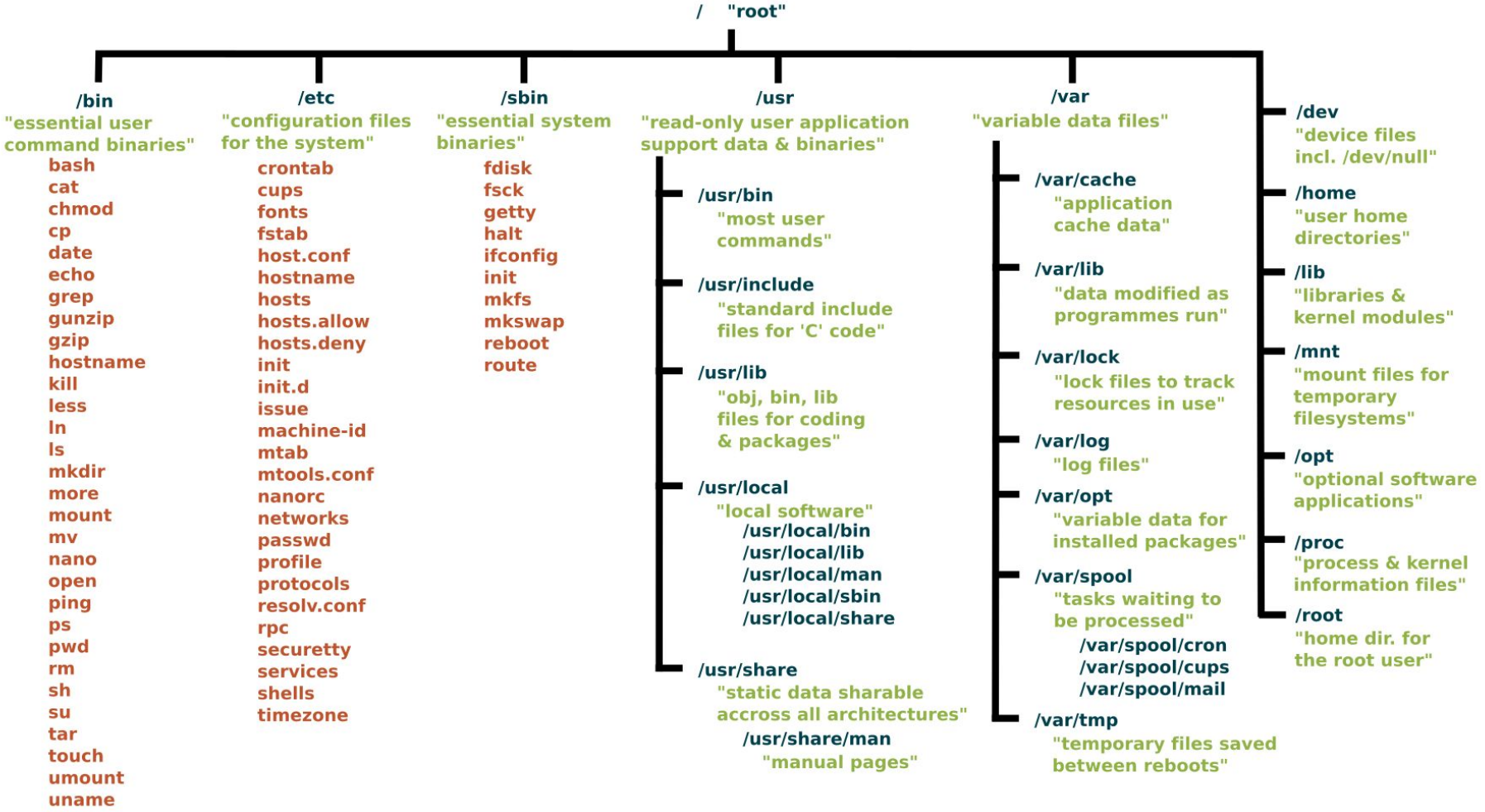
/sys is another directory you shouldn't really touch. It's used by the system to contain information about devices connected to your machine. This would store information like screen brightness.

Linux File System /tmp

/tmp contains temporary files, usually placed there by applications. This directory is also writable by any user, so a good place for hackers attacking a Linux system to download payloads.

Linux File System /var

/var is used to store variable files, or files that will change a lot. Logs are usually stored in /var/logs.


/ "root"
/bin
"essential user command binaries"
bash
cat
chmod
cp
date
echo
grep
gunzip
gzip
hostname
kill
less
ln
ls
mkdir
more
mount
mv
nano
open
ping
ps
pwd
rm
sh
su
tar
touch
umount
uname
/etc
"configuration files for the system"
crontab
cups
fonts
fstab
host.conf
hostname
hosts
hosts.allow
hosts.deny
init
init.d
issue
machine-id
mtab
mtools.conf
nanorc
networks
passwd
profile
protocols
resolv.conf
rpc
securetty
services
shells
timezone
/sbin
"essential system binaries"
fdisk
fsck
getty
halt
ifconfig
init
mkfs
mkswap
reboot
route
/usr
"read-only user application support data & binaries"
/usr/bin
"most user commands"
/usr/include
"standard include files for 'C' code"
/usr/lib
"obj, bin, lib files for coding & packages"
/usr/local
"local software"
/usr/local/bin
/usr/local/lib
/usr/local/man
/usr/local/sbin
/usr/local/share
/usr/share
"static data sharable accross all architectures"
/usr/share/man
"manual pages"
/var
"variable data files"
/var/cache
"application cache data"
/var/lib
"data modified as programmes run"
/var/lock
"lock files to track resources in use"
/var/log
"log files"
/var/opt
"variable data for installed packages"
/var/spool
"tasks waiting to be processed"
/var/spool/cron
/var/spool/cups
/var/spool/mail
/var/tmp
"temporary files saved between reboots"
/dev
"device files incl. /dev/null"
/home
"user home directories"
/lib
"libraries & kernel modules"
/mnt
"mount files for temporary filesystems"
/opt
"optional software applications"
/proc
"process & kernel information files"
/root
"home dir. for the root user"

Setting up a Linux Server

Setting up a Server

Install Ubuntu as a VM. You will not need a lot of resources (2 cores and 4GB of RAM with 20GB storage is plenty).

You can get Ubuntu from their website if it is not on Azure. It is downloaded by donation, so feel free to donate if you'd like. Otherwise, you can download it for free.

Set up a user, this will be our main user for configuring the Linux server.

SSH

SSH stands for **S**ecure **SH**ell. This is a communication protocol that allows one to remotely get a shell on a system. This is a very popular way to interact with Linux machines. Ideally, we would setup SSH and only use Linux's CLI (Command Line Interface).

By default, Ubuntu may have an SSH server installed already. If not, we can install it with the following command.

```
sudo apt install openssh-server
```

Use the following command to enable the SSH service. This will make sure SSH is started whenever the Linux machine boots up.

```
sudo systemctl enable ssh
```

You can then start the service with the following command.

```
sudo systemctl start ssh
```

You can check on the ssh service's status with this command. Press Ctrl + C when you are done (this is used as a user input to interrupt any command being run on Linux machines)

```
sudo systemctl status ssh
```

SSH

You can now SSH into your Ubuntu machine from your Windows host machine. Open CMD on your host machine and type the following command, replacing user with the username you set up during install (not your host's username) and the IP of your Ubuntu machine.

```
ssh <user>@<ubuntu_ip>
```

You will be prompted for a password. Once you enter it you will see your Linux CLI. Type "exit" when you are ready to leave.

Our configuration allows for brute-force attacks, because it relies on username/password for authentication. auxiliary/scanner/ssh/ssh_login on Metasploit can be used to brute-force your Ubuntu machine. Try it out!

SSH Hardening

Let’s harden our SSH server. For starters, let’s get rid of the username/password authentication. We can use keys to authenticate into an SSH server instead.

We will be creating a private key and public key for our host machine. We will store our Public key on our Ubuntu machine so that we can connect.

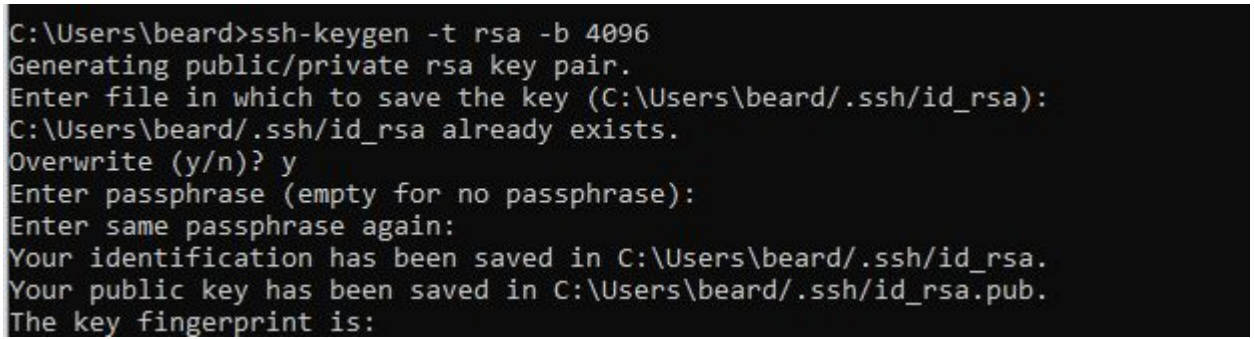
SSH Hardening - How keys work

Once we get our public key onto our Ubuntu machine, we can use it to try to authenticate to our Ubuntu machine. We sne dht public key over to the server, where it replies with an encrypted message. We can then use our private key to decrypt the message and send it back. This method works better than a username/password as it is much harder to brute-force. We can also add a password on our private key to further improve security in the case our private key is stolen.

SSH Hardening - Generating Keys

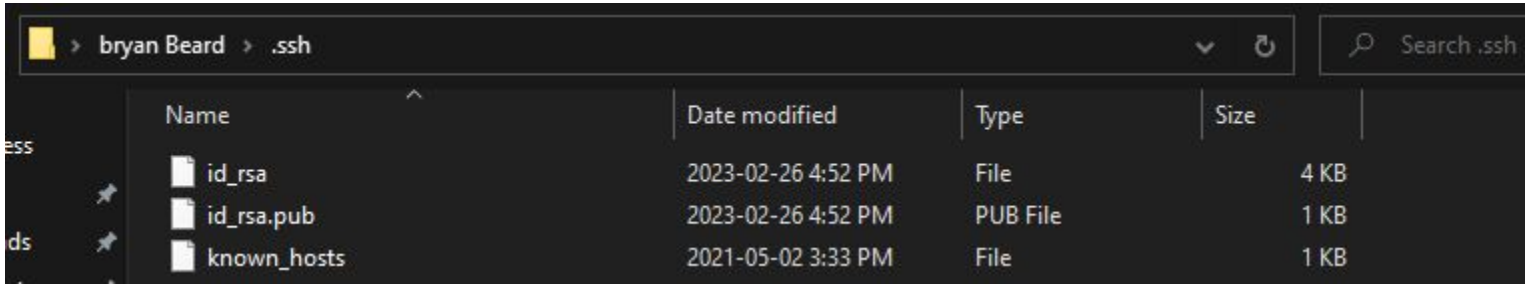
On our host machine, open CMD and type ssh-keygen -t rsa -b 4096. This will make an RSA 4096 key pair, which is considered a secure length by today's standards. It will prompt you where to save the files, where the default (.ssh/id_rsa) is fine, so just press enter.

You will then be prompted to add a passphrase for the key. Add a password, as pressing enter would leave the private key with no password. If our private key is stolen,and attacker can use it as much as they'd like without a password.



SSH Hardening - Generating Keys

Now your keys will be created in C:\Users\<username>\.ssh. Note that .ssh will be hidden because it has the . before it. You will manually have to type the file path in your File Explorer to see the keys.



SSH Hardening - Generating Keys

Right-click and open your id_rsa.pub with notepad by selecting “Open With”, and selecting “Notepad” from your list of apps. We will be transferring the contents of this file to your Ubuntu Server.

NEVER EVER share your private key.

SSH Hardening - Generating Keys

On your Ubuntu server, navigate to your home directory (/home/<username so mine is /home/bryan). And make a directory called .ssh.

```
mkdir .ssh
```

Navigate into .ssh

```
cd .ssh
```

Lastly, create a file called authorized_keys

```
nano authorized_keys
```

Paste your host's public key from Notepad into here and use Ctrl+x, Y, Enter to save the file. This is exiting Nano and saying yes to saving the document. You should now have a file in /home/<username>/.ssh/authorized_keys containing your host machine's public key.

SSH Hardening - Generating Keys

Now we need to configure the SSH server to only accept keys. This file is /etc/ssh/sshd_config

```
sudo nano /etc/ssh/sshd_config
```

You will see a lot of lines with a “#” in front of them. These are comments, and won’t be read by the system as a configuration.

Find the line (using the arrow keys) that says “#PubkeyAuthentication yes” and delete the “#”. This will uncomment the line and configure the server to allow public key authentication.

Find the line that says “#PasswordAuthentication yes” and uncomment it, changing the yes to a no. This will disable username/password authentication. Press Ctrl+x, y, Enter to save the file.

SSH Hardening - Generating Keys

We can restart the SSH service with `sudo systemctl restart ssh`

Whenever we make a change to the configuration of a service that is running, we will need to restart it.

Now when we ssh from our host machine using `ssh <user>@<ubuntu_ip>` , we will be prompted to enter the password for our private SSH key rather than the password for our user.

Additionally, if you try `ssh <user>@<ubuntu_ip>` from your Kali machine, you should be presented with an error saying "Permission denied (publickey)". This ensures that an attacker (your Kali machine) cannot SSH into your Ubuntu server without your private key and the password that goes along with it, stopping a brute-force attack.