`echo "I'M IN!!!"`

Using the command line for hacking

Breanne Boland
enterprise security engineer
breanneboland.com
twitter.com/breanneboland

whoami

- An appsec engineer with an ops/infrastructure background (which is why this stuff is important to me)
- A writer
- Someone who lives in Oakland and looks forward to seeing you all in person at the next DoS <3

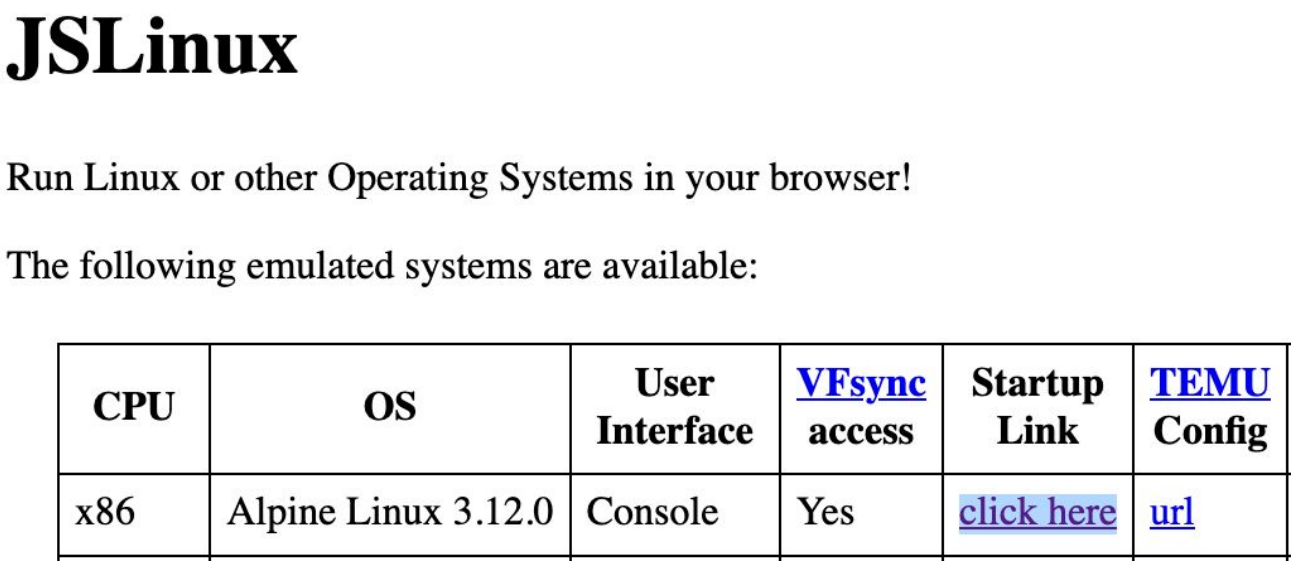
What are we going to do today?

We’re going to work through a series of commands and try them out. I’ll introduce one, explain it, and show a couple of examples. Then you can try it, and we can talk through any problems or questions that come up.

Before I talk some more...

https://bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192

Or: https://bellard.org/jslinux/ and select this:



JSLinux

Run Linux or other Operating Systems in your browser!

The following emulated systems are available:

CPU	OS	User Interface	VFsync access	Startup Link	TEMU Config
x86	Alpine Linux 3.12.0	Console	Yes	click here	url

And...

Clone this repo:

https://github.com/hellocerulean/DoS2021

(It has notes, links, and other goodies.)

Why learn command line stuff?

- Some things are just faster to do this way than via a graphical user interface (GUI; think a browser, or moving folders and windows around)
- If you work with Linux servers, you will mostly be dealing with the command line
- You look like a super-cool hacker
- Being better at this stuff helps with literally every engineering job, security or otherwise
- A CTO at my previous job: “You will never be sorry to know more Bash.” True!

What have you done with the command line before?

What are you hoping to learn?

And what kind of system are you working with today?

Answer in chat!

Different flavors of command lines

- Linux and Mac CLIs and how they differ
- If in doubt: man page, please
- Flavors of Linux
- What’s most likely to come up in this work

What’s not on the menu

- PowerShell and Windows concerns
- zsh and fsh, other shells, and their differences

Did you clone the repo?

https://github.com/hellocerulean/DoS2021

React when you've successfully cloned it, please!

Vocabulary, briefly

Server == box == machine… but I will try to only say server, I promise.

$thing is Bash notation for a variable. I use it to mean “add your own needed term here.”

Bash is usually the term used for the language. The command line or shell is what we’re typing into. These terms sometimes get used in overlapping ways, though.

In these slides, if something is a command, it looks like `this`.

That means it’s something you can paste into the command line and have it mean something. If it’s a key (spacebar, escape, ctrl), it’s in regular font.

Other useful resources

- Man[ual] pages (including `man bash`)
  - Search within a man page with `/$searchTerm`
- explainshell.com
- Google “best $command commands” - people LOVE writing useful posts about this stuff
  - Put things in explainshell before you run them on your own machine, please.
- The trouble with googling “bash >” and how to get around it
  - symbolhound.com can help too

A few command line niceties

- Control-C (or: omg the scrolling is going to set my computer on fire)
- `sudo !!` (and sudo generally)


me
Permission denied
sudo

A few command line niceties

- Control-C (or: omg the scrolling is going to set my computer on fire)
- `sudo !!` (and sudo generally)
- Tab completion
- Beware of curly/smart quotes. Nice in typography, possible nightmares on the command line
- . can usually be used to mean “this directory.” Think of it as shorthand for “here.”

Anatomy of a command

- The command itself

ls

Anatomy of a command

- The command itself
- Flags

```
ls -al
```

Anatomy of a command

- The command itself
- Flags
- Arguments

```
ls -al ./dir/subdir
    ls -al /lib
ls /lib /dir /cat_pix
```

Anatomy of a command

- The command itself
- Flags
- Arguments
- Piping, briefly (more later)

```
ls -al . | grep DoS
```

Anatomy of a command

- The command itself
- Flags
- Arguments
- Piping, briefly (more later)
- `--help` and `-h`

```
ls --help
```

What do you know about `ls` now?

`ls` means list files

`ls` can take both absolute and relative paths as arguments (including both at once!)

`ls` has lots of flags, and you can layer them all on after a dash

`ls --help` will show you so many options

Nice! Let’s learn about servers from our command lines.

How do I learn about the server? Or: fun with OSINT

- `nmap -F 8.8.8.8` (or any IP; try on your own computer)
  - `ping $URL` to get IP
- `curl -i $url`
- `whois` and `dig`

How do I connect to the server?

- Why connect to a server at all?
- `ssh` and its history
- `ssh -i ~/.ssh/your_key.pem`
  - My example today: `ssh -i ~/.ssh/breanne_dos.pem ec2-user@34.218.206.14`
- The joys of `-v`, `-vv`, and `-vvv`

Let’s break that down

`ssh`: the command

`-i ~/.ssh/breanne_dos.pem`: -i says I’m using an identity (key) file, the path is where the file is

`ec2-user`: the Linux user I’m logging in as. This is standard AWS stuff.

`@34.218.206.14`: the public IP of my server

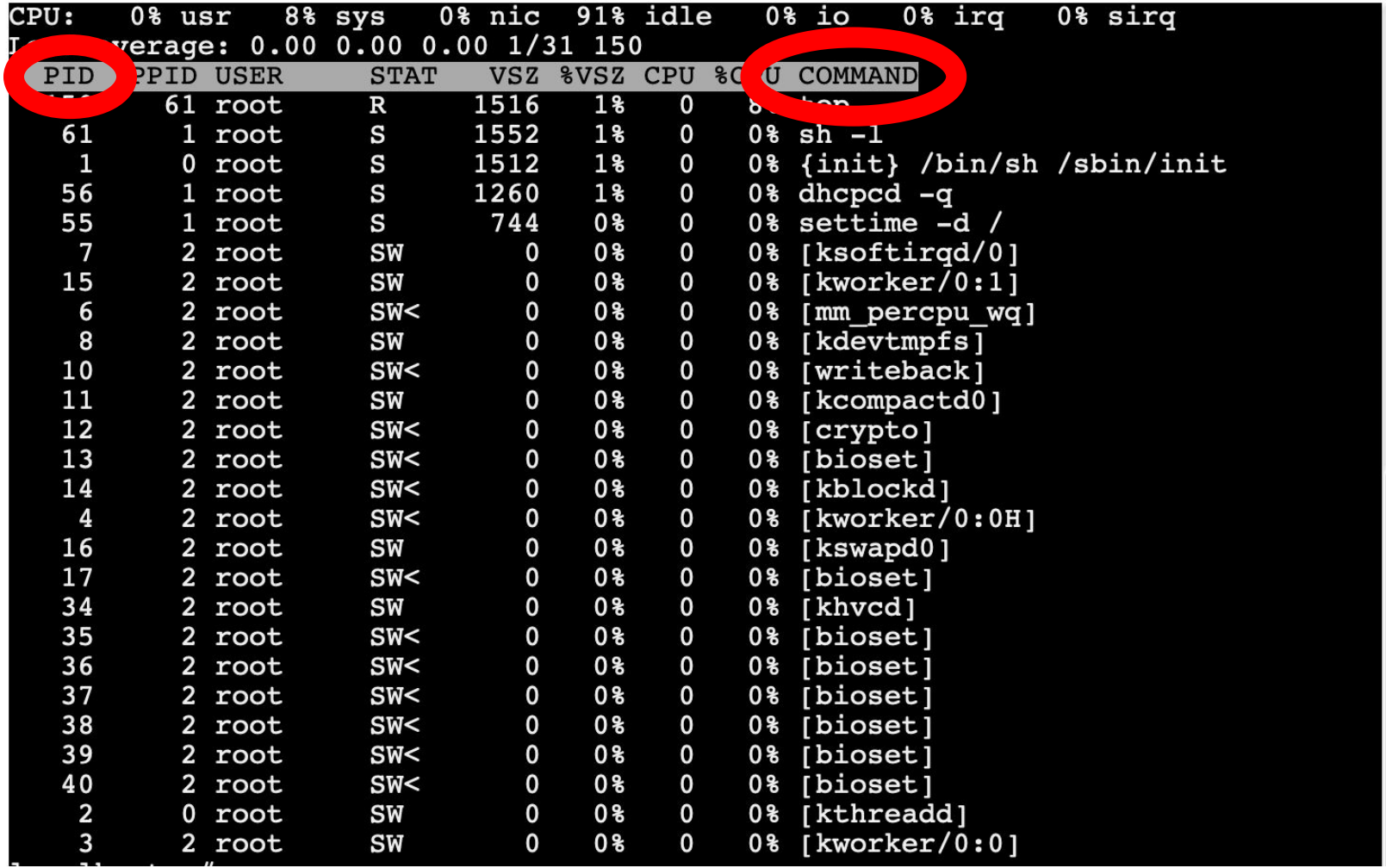
How do I navigate this server?

- `ls` again!
- `cd /path/to/something`, `cd ..`, and `cd --` too
- `history` or ctrl-r for reverse-i-search
- `whoami` and `pwd`
- `cat $file` to print a file's contents to the command line

How do I see what's going on in the server?

`top`

Type `q` to exit - this works for lots of command line programs that take over the whole window.


CPU: 0% usr 8% sys 0% nic 91% idle 0% io 0% irq 0% sirq
Load average: 0.00 0.00 0.00 1/31 150
PID PPID USER STAT VSZ %VSZ CPU %CPU COMMAND
61 root R 1516 1% 0 0% sh -l
1 0 root S 1512 1% 0 0% {init} /bin/sh /sbin/init
56 1 root S 1260 1% 0 0% dhcpcd -q
55 1 root S 744 0% 0 0% settime -d /
7 2 root SW 0 0% 0 0% [ksoftirqd/0]
15 2 root SW 0 0% 0 0% [kworker/0:1]
6 2 root SW< 0 0% 0 0% [mm_percpu_wq]
8 2 root SW 0 0% 0 0% [kdevtmpfs]
10 2 root SW< 0 0% 0 0% [writeback]
11 2 root SW 0 0% 0 0% [kcompactd0]
12 2 root SW< 0 0% 0 0% [crypto]
13 2 root SW< 0 0% 0 0% [bioset]
14 2 root SW< 0 0% 0 0% [kblockd]
4 2 root SW< 0 0% 0 0% [kworker/0:0H]
16 2 root SW 0 0% 0 0% [kswapd0]
17 2 root SW< 0 0% 0 0% [bioset]
34 2 root SW 0 0% 0 0% [khvcd]
35 2 root SW< 0 0% 0 0% [bioset]
36 2 root SW< 0 0% 0 0% [bioset]
37 2 root SW< 0 0% 0 0% [bioset]
38 2 root SW< 0 0% 0 0% [bioset]
39 2 root SW< 0 0% 0 0% [bioset]
40 2 root SW< 0 0% 0 0% [bioset]
2 0 root SW 0 0% 0 0% [kthreadd]
3 2 root SW 0 0% 0 0% [kworker/0:0]

How do I see what's going on in the server?

`top`

Type `q` to exit - this works for lots of command line programs that take over the whole window.

`ps aux`: also good, but doesn't update live (but can be `grep`ped!)

`lscpu`

How do I affect what's going on in the server?

- Altering processes, including `kill`
  - `killall` if you have the process name
  - `kill` if you have the PID
  - `kill -15` for a graceful termination
  - `kill -9` for PROCESS DEAD THIS INSTANT PLEASE
- Running your own code
  - `python3 your_script.py`

How do I navigate this big log file?

- `less $filename [$otherFilename $anotherFilename]`
  - Scroll or arrow to go up or down
  - Spacebar or `f` to go forward one page
  - `b` to go backward one page
  - `/$pattern` to search forward, `?$pattern` to search backward
  - `q` to quit (there it is again!)
  - SO MANY MORE
- `tail -f $filename` to have it show additions to the file in real time (great for active logs)

How do I find a needle in this ~~haystack~~ big log file?

`grep`!

Three lines of context before and after: `grep -B 3 -A 3 “$word” $file`

Combine `history` with `grep`: `history | grep $command`

How do I find a needle in this ~~haystack~~ big log file?

`grep`!

Three lines of context before and after: `grep -B 3 -A 3 "$word" $file`

Combine `history` with `grep`: `history | grep $command`

`grep` challenge!

- What's Breanne's favorite line in *Anne of Green Gables*?
- What does Breanne think is one of the funniest scenes in *Anne of Green Gables*?
- What does Breanne think is the most hilarious scene in the movie adaptations of *Pride and Prejudice*?
- What part of *Pride and Prejudice* does Breanne think is the saddest but also one of the most true?

Pipes!

- Piping to `grep`
  - `cat $file | grep $searchTerm`
- Piping to `grep` and then `wc`
  - `cat $file | grep $searchTerm | wc -l`
- Piping to `head` and `tail`
  - `cat $file | head -5 | tail -5`

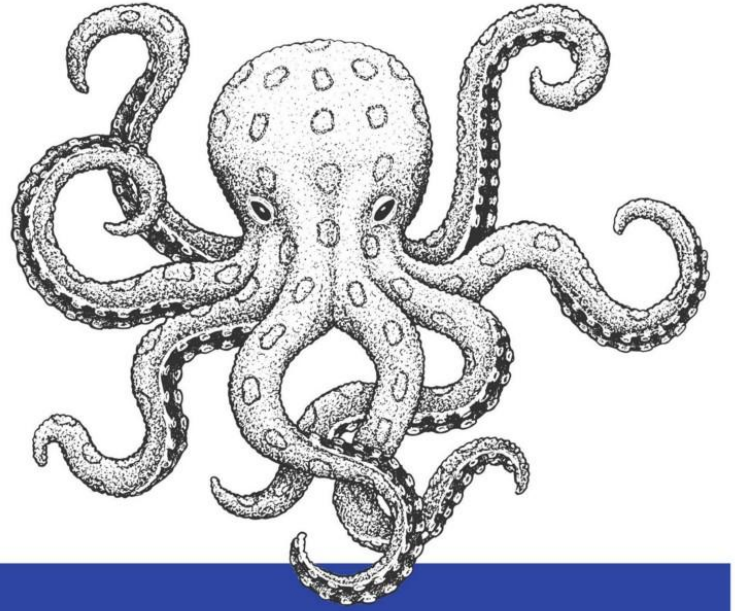
> and >>

- >> is append (append == a friend)
  - `cat logfile.txt >> otherlogfile.txt`
  - This puts the text of logfile.txt at the end of otherlogfile.txt, preserving the second file's original contents.
- > is overwrite (DANGER, DANGER)
  - `cat logfile.txt > otherlogfile.txt`
  - This obliterates the previous contents of otherlogfile.txt, overwriting them with logfile.txt. If otherlogfile.txt doesn't exist yet, it creates it.
- IF IN DOUBT: >>

How do I cover my tracks or otherwise change things?

- Altering logs :)
  - /var/log/auth.log
  - /var/log/secure
  - /var/log/messages
- How to delete history
  - `history -d $lineNumber` to take out just one line
  - `history -c` to nuke the whole thing from orbit (not a subtle move, but it can work)
- Edit files with vi
  - `vi $filename` to edit (`touch $filename` to create)
  - Type `i` to type (think insert)
  - Type escape + `:x` to save and exit (NOW YOU KNOW THE SECRET)
  - Type escape + `:q` to exit without saving
  - That's enough on that for today, I promise

Just memorize these fourteen contextually dependant instructions



Exiting Vim

Eventually

O RLY?

@ThePracticalDev

Resources for continuing command line excellence

- Vim Adventures
- `man bash`
- My first Bash tutorial
- Codecademy for Bash
- Exercism for Bash
- Bash cheatsheet
- Pacific Hackers on OSINT
- More on `less`
- Julia Evans on Linux and a command line zine
- $PATH and system variables

What’s one thing you learned that you’re excited about?

Questions?

We can talk command line, appsec jobs, hiring, job transitions, or whatever else you want to know about.

twitter.com/breanneboland

https://breanneboland.com/blog/2021/03/22/day-of-shecurity-2021-intro-to-command-line-awesomeness/