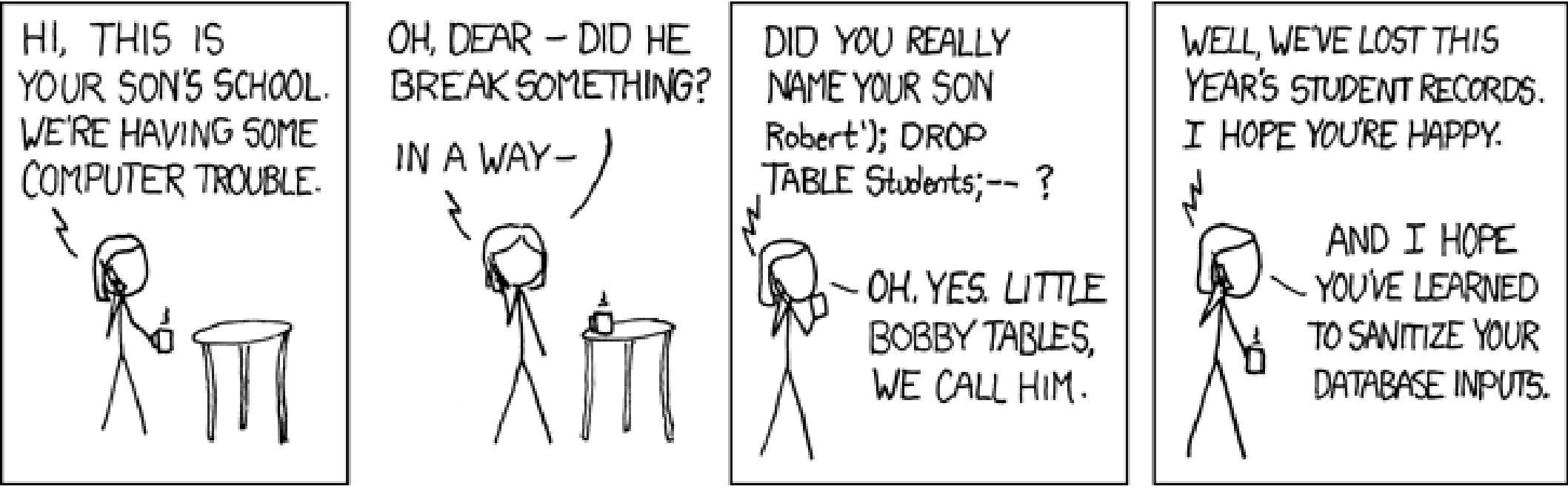
CSCI 3330 Cybersecurity



Word of the day

MODULO prep. Except for. An overgeneralization of mathematical terminology. “Well, LISP seems to work okay now, modulo that GC bug.” “I feel fine today modulo a slight headache.”

Command line

- `printf '%d %d == %d\n' 1 2 3`
- `... steps ...`
- `main() with argv:`

  `{“printf”, “%d %d == %d\n”, “1”, “2”, “3”}`
- `... steps ...`
- `1 2 == 3`

Shell constructs argv

- Backslashes and quoting
  - → Files with *spaces*
- `execv(argv)`

main(): conventions

- `ls -l -a == ls -la`
- `wget -O page.html mysite.com`
  `== wget -Opage.html mysite.com`
- `ls -l -l -l == ls -lll == ls -l`
- `wget -Ox.html -Oy.html mysite.com`
  `== wget -Oy.html mysite.com`

main(): conventions

- How do we pass arguments that look like options?
- `ls -- -l` → lists directory named “-l”
- Alternatively for paths: `ls ./-l`
- But this is less useful for variables that are absolute paths or that are not paths at all

getopt()

- Common function on Unix systems for parsing arguments
- C/python/bash/etc.

Long options (GNU)

- `ls -a == ls --all`
- `ls --sort time == ls --sort=time`

Many commands behave differently

- echo
- gcc
- find
- ffmpeg

Shell script variables

- Word splitting
- `var="foo bar bletch"`
- `ls $var` → argv: {“ls”, “foo”, “bar”, “bletch”}
- `ls "$var"` → argv: {“ls”, “foo bar bletch”}

In-class exercise

1)Create (touch) a file named “foo bar bletch”

2)Create a file named “-h”

3)Again with another approach

4)Create a file named “- -” [hyphen space hyphen]

5)Again with another approach

6)Echo your home directory ($HOME)

7)Echo the string “-e”

Finally

Type:

```
FOO='--help -h'
```

How can you create a file named “--help -h” using `touch` and referencing `$FOO`?

Final final challenge: `rm` these files

Handling user input (sh)

- download.sh:

```
wget -O page.html $1 ❌❌❌
```

Handling user input (sh)

- download.sh:

```
wget -O page.html $1 ❌❌❌
```

Handling user input (sh)

- download.sh:

```
wget -O page.html $1
```

❌❌❌

- What if argv[1] ==
  “-O /etc/shadow something”?

Handling user input (sh)

- download.sh:

```
wget -O page.html $1
```

❌❌❌

- What if argv[1] == "-O /etc/shadow barf.com"?

```
wget -O page.html -O /etc/shadow barf.com
```


IT'S FINE

Handling user input (sh)

- download.sh:

```
wget -O page.html $1 ❌❌❌
wget -O page.html "$1" ✅✅✅
```

- C: `system()`
- python: `os.system()`
- Execute string as if you had typed it into your shell
- `os.system('echo $USER')` → jeffrey

Handling user input (py)

- download.py

```
os.system("wget -O page.html " + url)
```

Handling user input (py)

- download.py

```
os.system("wget -O page.html " + url)
```

Handling user input (py)

- download.py

```
os.system("wget -O page.html " + url)
```

❌❌❌

- What if url ==
  “-O /etc/shadow something.com”?

Handling user input (py)

- download.py

```
os.system("wget -O page.html " + url)
```

❌❌❌

- What if url == “barf.com; rm -rf /”?

```
os.system("wget -O page.html barf.com;
           rm -rf /")
```


IT'S FINE

Fixing this one

- Does same fix work?

Fixing this one

- Does same fix work?
- There exists `shlex.quote()`
- ```
  os.system("wget -O page.html " +
                shlex.quote(url))
  ```
- Problems:
  - Tedious and error prone
  - How does python know what shell you're using?
  ```

Remove the in-band signal!

- os.exec*() family: replace current process image with another
- subprocess module: manage child process

Handling user input (py)

- download.py

```
os.system("wget -O page.html " + url)
```
❌❌❌

```
os.execv("wget",
         ["wget", "-O", "page.html", url])
```
✅✅✅

In-band signaling eliminated!

C also has exec*() functions.

Databases

- You could take an entire class on these…
- “Relational” databases
- SQL

Basic idea

- Tables
- Columns: mostly fixed (part of your schema)
- Rows: data entries (get added over time)
- These rows can be “related” (you can reference them from other rows in same/other tables)

SQL

- ***C***reate: INSERT
- ***R***ead: SELECT
- ***U***pdate: UPDATE
- ***D***elete: DELETE
- In SQL, -- begins a line comment

Demo

SQL in Web apps

- Large, dynamic data
- Arbitrary user input

```
"DELETE FROM stuff WHERE id =  ' "  + input + " ' "
```

Assignment 1

- Deadline: September 18, 2025, 10:00pm
- Permissions, entropy, randomness
- https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment1.pdf

Assignment 2

- Deadline: September 25, 2025, 10:00pm
- Setuid binaries, directory traversal, command line injection https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment1.pdf