Cybersecurity
And Why You Should Study It!
Allie :)
Confidentiality

Shhh… can you keep a secret?

All data should be kept private at all times! You need to be super careful about who has access to what (i.e. who is authorized to see the data).

Most common attack due to weak confidentiality:

Man-in-the-Middle Attacks

Man in the Middle Attacks

Uh oh, someone is eavesdropping on you!

- When a bad guy positions themselves in a private communication between a user and an application.
- Goal: steal personal information (passwords, credit card numbers, social security numbers)
- Tricky because skilled hackers can make it seem like nothing is wrong (they can hide really well)

Basically: You send some hot gossip in the mail, and your mailman opens the envelope, reads the information, then resends it and nobody will ever know!

Integrity

Don’t let anyone mess with your data!

In addition to being private, data needs to be trustworthy! Integrity of data is upheld if the data is accurate and reliable (basically, nobody can get in the system and change it without permission)

How to make sure attackers can’t compromise data integrity? Hide your data! Fancy techniques like encryption, hashing, and more can protect your data.
What Can Go Wrong with Data Integrity?

Instead of one specific "main attack," problems with data integrity arise from:

**Problems with authentication and authorization**

If someone gets into the system but they're lying about their identity

...very bad :(

**Availability**

Ok now my data is super secret and private but wait... now who gets to see it?

Data isn't much good to a company if nobody can access it; the tricky part of cybersecurity is making sure that people are actually able to access the top secret data.

Systems need to work! If a system is compromised and there is no recovery plan, data can get lost, or at least take a long time to access (making customers very very angry).

What if it's compromised on purpose... introducing the Denial of Service (DoS) attack

**Denial of Service (DoS) Attacks**

Uh oh, someone wants to make it hard to get your data!

- Hacker spam a server with traffic
- Special case: Distributed DoS attack: hacker uses multiple computers to flood the target
- Overflows the server and interrupts the service being provided

CS15 example: GPT4 uses rate limiting: you can only enter queries every 15 seconds, so the server doesn't get overloaded
What can you do about it

Three main pathways:
1. Policy (non-technical)
2. Blue Team (technical defense)
3. Red Team (technical offense)

Non-technical Cybersecurity Roles
- Names like “Policy Writer”, “Governance and Risk”
- Great for people who care about security but don’t love the technical aspect
- Security engineers follow rules to secure their systems — you can write the rulebook!
- Skills required: security awareness, writing skills, critical thinking, top-level understanding of technology

Blue Team
- Technical defensive role
- Responsible for securing and protecting systems
- Planning design a secure system
- Threat protection: always be alert!
- Roles like “Application Security”, “Cloud Security”
- Great for people interested in coding and how computer systems work
- High-projected salaries (very high)
Red Team

- Technical, offensive role
- Blue Team works hard to secure the system. Red Team tries to break it.
- Blue Team runs various tests to find weak points to fix.
- Roles like "Penetration Testing"
- Great for people interested in coding and who like to break things!
- High projected salaries.

Interested in learning more?

There are some awesome security classes in the Brown CS department:
- CS22: Learn the math behind encryption (you should all take it!)
- CS23: Computer systems are super important in security (prerequisite for other security classes also)
- CS1400: Cryptography (how to secure online communication)
- CS1650: Software Security and Exploration (hacking into the system)
- CS1660: Intro to Computer Systems Security
- CS1510: Intro to Cryptography and Computer Security
- CS1515: Applied Cryptography

The Battle of Marathon
490 BCE
Athens population: 315,000  
(about 0.3% of the world population)

Persian Empire population: 50,000,000  
(about 50% of the world population)

Who wins?

Athens
But how??

What is a Shell?

- User interface for operating system commands
- Shells:
  - Graphical User Interface (GUI)
  - Command Line Interface (CLI)
- Goal to create an effective workflow to accomplish certain tasks

CLI (Command Line Interface)

- The main graphical component is a window that allows you to enter text input
- User types text commands the program will run
  - ex. `echo hello`
- ex. Terminal
- Shell-specific scripting languages
  - `ex. bash, zsh`
CLI: Good Idea or Bad Idea?

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster</td>
<td>GUI takes a lot to load</td>
</tr>
<tr>
<td></td>
<td>Practice provides fluidity</td>
</tr>
<tr>
<td>Accessible</td>
<td></td>
</tr>
<tr>
<td>Learning curve</td>
<td>Reduced interaction for user</td>
</tr>
<tr>
<td></td>
<td>No clicking or dragging</td>
</tr>
<tr>
<td></td>
<td>Aesthetics = whack</td>
</tr>
</tbody>
</table>

Common Commands

- `cd` - change directory (type `cd` then use tab key to autocomplete)
- `ls` - list directory contents
- `touch` - create a new file or "touch" existing file
- `rm` - remove a file
- `mkdir` - create a directory
- `rmdir` - remove a directory
- `mv` - move one file to another location
- `cp` - copy one file
- `cat` - print the contents of a file
- `grep` - search for text within files
- `clear` - clear the terminal

Flags

- `java` - Java -version
- `ls` - list
- `rm` - remove
- `chmod` - change file permissions
- `grep` - search for text
- `clear` - clear the terminal

Flags (Mac only but there are Windows equivalents)

- `open` - open a file
Chaining Commands

- **Piping**
  - Redirects the output of one command into the input of another
  - `command_1 | command_2 | command_3 | ... | command_N`
  - Ex: `ls -a | wc -l` passes the output of `ls` into `wc`

- **&& (AND)**
  - Chains a sequence of commands together
  - `command_1 && command_2 && command_3 && ... && command_N`
  - Ex: `cd src && ls` will go into src directory and list all contents

Useful Tips and Tricks

- Retyping a recent command?
  - Press the up (and down) arrow to navigate history
  - `Control + R` to do a back-history search

- Can’t remember the filepath or command?
  - Press Tab once for autocomplete, twice for a list of everything in that directory

- Want to stop something you accidentally ran?
  - `Ctrl + C` to interrupt the process
  - `Ctrl + Z` to pause a process
  - Can then send to foreground (`fg`) or background (`bg`) using the process ID

- Wish you had a shortcut for a long command?
  - Create an alias in `.zshrc` or `.bash_profile`
  - Ex: `alias compile_run_pong="cd pong && javac *.java && cd .. && java pong.App"`
  - open ~/.zshrc → type your alias (save file) → refresh your terminal → enjoy
-FUN- Commands

- date
- ps <text>
- figlet <text>
- ns
- telnet towel.blinkenlights.nl
- erase <text>
- fortune
- cowsay <text>
- asdf

*Note: for some of these commands, download may be required (Google is your friend)*

Text Editors in the Terminal

- nano
- vim
- emacs

Interested?

Introduction to Computer Systems
When your simulations become more real than reality

Raster graphics, used in CS230, work with a fixed number of colored pixels:
- 2D arrays, anyone?
- Manually compute the value of each pixel according to what you want to see.
- Used for texture mapping or draw your own filter and transform images.

Beyond JavaFX
Beyond JavaFX

- Raster graphics, used in CS230, work with a fixed number of colored pixels
- 2D arrays, anyone?
- Manually compute the value of each pixel according to what you want to see
- Used to simulate scenes or draw your own filters and transform images

Simulating Our World

Let there be light!

How do we simulate our world on a screen?

- BIG IDEA: use mathematical / physical laws of motion to describe the movement and appearance of the objects you code
- Using gravity + collision detection in Doodle Jump
- You didn’t have to hardcode where exactly the doodle would be at every time step
- That would have made your program WAY too long and impossible to play!
Raytracing + Pathtracing
- Describe a scene and have a computer render it “photorealistically.”
- In real life, things appear the way they do because of the way light interacts with them.
- Follow light rays from a light source and figure out how much light hits the spot in the image, and where the lights bounce off the objects.

Which one is real?

Real World Features On Your Screen
- Not everything we want to render is a solid object; we also want real life features (like water) with realistic interactions.
- We're able to bring these features to life through these simulations, and they're only improving from here.
- We use math and clever coding techniques to simulate real-life objects digitally.
- Splashes, ripples, reflections, buoyancy of the ball, etc.
How can we use AI to optimize creativity?

- Creation takes time!
  - Animation: a drawing for every frame, ~24 frames/second
  - A lot faster to visualize an image mentally than to put "on paper!"
  - Right: timelapse of 30 hours of work for 8 comic pages

The good, the bad, and the ugly

- AI trained off of stolen datasets
  - Big ethical debate
- On the other hand, creation has been "gatekept"
  - Art not known as a lucrative field
  - Making some projects requires a huge amount of funding or a big studio
- Not going away anytime soon, so how can we creatives harness it to work for us and not against?
Silly Premise: Andy's X-Men

- We want to make a cover for comic, Andy's X-Men.
- We have a sketch, but we want to make polished concept art. Can AI help?
  - yes! (to an extent)

Midjourney AI: Powerful, but unwieldy

- AI tools can create powerful images, but they can be challenging to use.
- Users need to understand the limitations and capabilities of the technology.
- Collaboration between artists and AI can lead to exciting results.

49

50

51
Infinite Variations!

- Great for brainstorming
- Less great for polishing existing concepts

Render photorealistically
Example use case: In-Betweens ("Tweening") & Rotoscoping

- **Tweening:** Animations are composed of key frames (like specific poses), and in-betweens, which are frames between keyframes to make motion smooth. Tweening is the process of drawing the in-between frames.
- **Rotoscoping:** Drawing over video to make an animation.
  - Classic Disney movies like The Little Mermaid were made this way!
- Can be tedious and repetitive

More Graphics:

- Here are some resources:
  - Two minute papers youtube channel (highly highly recommend, even for non-graphics related videos)
  - Pixar graphics library
- Here are some classes you could take:
  - CS1230: Introduction to Computer Graphics (requires CS220)
  - CS220: Introduction to Computer Animation
  - CS224d: Interactive Computer Graphics (requires CS220)
Industry

Of course we do. That's why I got into CS in the first place.

So you want to get a job?

What even is "industry"?

Government

Non Profit

For Profit

Education Tech ("EdTech")

Health Tech

Financial Tech ("FinTech")

Consumer Tech

Advertising

AI Tech

Bio Tech

Ad Tech

VR and Augmented Reality

Security

Databases

Communication

Lots and Lots More…

Coursera, Khan Academy, Duolingo, Canvas

AthenaHealth, HealthCare.gov, Clover, Flatiron

Stripe, Jane Street, Square, Bloomberg

Google, Facebook, Apple, Dropbox

Pixar, Dreamworks

Activision, Blizzard, Bungie, EA, Valve

OpenAI

Novo Nordisk, Moderna Inc., BioNTech

…

…

…

…
Types of Careers: SWE

- Software Engineer/Developer (often called SWE)
  - Focus on creating and coding the software
  - Variety of specialties: Test/Quality Assurance (QA), etc.
  - Not a code monkey, not coding 10-12 hours a day
  - Often in meetings collaborating on design, setting requirements, and talking to prospective customers
  - Depends on company/job, so research/ask about it during process
  - Can work on different parts of applications
  - Specialists: Frontend, Backend, Databases
  - Generalist: "Full-Stack"

- Focus on creating and coding the software

- Varieties of specialties: Test/Quality Assurance (QA), etc.

- Not a code monkey, not coding 10-12 hours a day

- Often in meetings collaborating on design, setting requirements, and talking to prospective customers

- Depends on company/job, so research/ask about it during process

- Can work on different parts of applications

- Specialists: Frontend, Backend, Databases

- Generalist: "Full-Stack"

Types of Careers: PM

- Project/Program/Product Manager
  - Some of our best HTAs have gone into Program Management
  - Focus defining what the product should be and what features it should have
  - Includes some level of project management/collaboration
  - Work with both prospective users and software developers
  - Technical position
  - Some PMs code and make prototypes
  - Can’t just tell everyone what to do. Have to convince the engineers that your plans are the best for the product
  - Being a PM doesn’t mean you can’t be a SWE after (and vice versa)

Other Types of Careers

- UX (User Experience) Designer
- UI (User Interface) Designer
- UX vs. UI
- Data Scientist
- Systems Programmer
- IT Architect
- And many more!
The Road to Jobs/Internships*

- Research companies
- Apply (online, at Tech Fair, at Career Fair, by email, etc.)
- Online assessment (usually asynchronous)
- Technical interview (Phone/Zoom/Google Hangouts)
- Onsite interview
- Offer

*Disclaimer: This may vary by company, some might have more/less steps.

Online Coding Assessments

- Many software development jobs & internships nowadays require some form of an online coding challenge/assessment to weed out most candidates.
- You code on an IDE of their choosing, interviewer gives some problem that you have to solve.
- If written in Java, they expect you to reason through different ideas and write code or pseudocode for a solution.
- If coding, they normally use whatever language you are most comfortable in.
- Problems are often algorithmic and/or involve some sort of data structure:
  - How could you reverse a LinkedList?
  - How could you build a Queue using two Stacks?
  - Imagine a scenario about some hypothetical game. How would you account for a specific case or rule?
- Many of the foundational data structures and algorithms needed for technical interviews are covered in CS200.

Technical Interviews

- Similar to online coding assessments.
- On an online coding pad or on whiteboard, interviewer gives some problem that you have to solve.
- You reason through different ideas and write code or pseudocode for a solution.
- Expect to talk aloud and show your work.
- Most importantly, they just want to see how you think.
- Problems often require algorithms and involve some sort of data structure.
What might an internship look like?

Working at a small company/start up
- < 50 people
  - Typically only designing 1-2 products
  - Small user base → take more risks in project features
  - Have a larger say in the direction of the project
  - Fast paced → push out features as fast as you can build them (every couple of days)

Working at a mid level - large company
- > 200 people
  - Building many products, can get exposed to multiple different technologies in one company
  - Larger user base → take less risks, work is heavily reviewed
  - Might not have as much freedom, but affect many more users
  - Slower paced → features pushed out every couple of weeks/months

What I Wish I Knew About CS Earlier...

& Why You Belong Here

Sarah

Only Some of the Resources at Brown

- WiCS - Women in Computer Science
  - Mentorship program, meetings and events
  - Supports the Artemis Project, a free summer camp for rising 6th-grade girls from the Providence area who show interest in science and technology.
- Mosaic
  - Advocate for diversity within Brown's CS community
  - Big little system, workshops, group study
- The UTA Program
  - Has changed my life!
There is more to CS than SWE

In fact, there are jobs out there that you and I have never even heard of...

Random CS Internships I’ve Explored

- Computer Science Teaching
- Product Management/Technical Project Management
- Technical Writing/Documentation Team
- Technical Specialist for Law Firm

And there are so many more! (UX Designer, Systems Architect, Database Administrator, Healthcare AI Developer, Quant Analysts)

“Everyone in this country should learn how to program because it teaches you how to think”

– Steve Jobs.
So... what makes a good programmer?
The TAs are not better programmers than you. We just have more experience.

Problems become familiar problems.
"I did something wrong..." "Now it doesn't work..."

"Where have I seen this before?"
"How can I figure out what happened?"
"Where can I find more information?"

Struggling + Frustration → Learning
You're learning how to learn.

The Goal of CS

Know how to code everything.

Know how to figure out how to do what you want to do.
Part of that process includes not knowing what to do. AND THAT'S COMPLETELY FINE.

What makes a good programmer?

- Knows how to write code,
- Learns and masters relevant tools,
- Persistent against stupid computers,
- Always willing to learn,
- And a pro at a little extra googling.

You're already on your way. Go get 'em.
Announcements

Apply to TA CS15 for next fall!

How do I apply?
- Applications for next fall will come out in early/mid March
- Application is short and non-binding
- We'll send an email to the whole course when applications are out!
Why should I apply?
- Participate in the skits!
- Make friends for life!
- Master Java and OOP!
- Mentor new CS students!
- Improve the course!
Lastly, we present... the best of the commit messages
You had a way with words...

Sometimes GitHub is hard...

We've seen you at your highest highs...
...and your lowest lows

We love you too <3

Yes 😊