# A Whirlwind Tour

Matthew Trost

CS107e Computer Systems from the Ground Up

February 28, 2022

Stanford University

# A Whirlwind Tour

Exposure to important topics…
…with some fun intermissions

- Memory Hierarchy
- Trust and Safety
- Memory Safety
- Where to Go From Here

# A Whirlwind Tour

Exposure to important topics…
…with some fun intermissions

- **Memory Hierarchy**
- Trust and Safety
- Memory Safety
- Where to Go from Here

# Where can a computer store data?

# Where can a computer store data?

- Registers
- Cache ← we'll talk about this
- RAM
- Flash / SSD
- Magnetic Disk
- Other computers (via network)
- Magnetic Tape

# Latency Numbers Every Programmer Should Know

```
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

Assuming ~1GB/sec SSD

Numbers from 2012. Jeff Dean; https://gist.github.com/hellerbarde/2843375

# Multiplying by a Billion…

| | | |
|---|---|---|
| L1 Cache Hit | 0.5s | One heartbeat |
| L2 Cache Hit | 7s | Long yawn |
| | | |
| Main Memory Reference | 100s | Brushing your teeth |
| | | |
| SSD random read | 1.7 days | Weekend |
| Read 1MB sequentially from memory | 2.9 days | Long weekend |
| Disk Seek | 16.5 weeks | 1.5 quarters at Stanford |
| Read 1MB seq. from disk | 7.8 months | |

2012 numbers. Jeff Dean; https://gist.github.com/hellerbarde/2843375

# OK, so what's our strategy?

- Really big caches?

# OK, so what's our strategy?

- Really big caches? **Too expensive…**

| Technology | Access Time | Price per GB |
|---|---|---|
| SRAM semiconductor | 0.5-2.5ns | $300 |
| DRAM semiconductor | 50-70ns | $6 |
| Flash semiconductor | 5,000-50,000ns | $0.40 |
| Magnetic disk | 5,000,000-20,000,000ns | $0.02 |
| Magnetic tape | - | $0.008 |

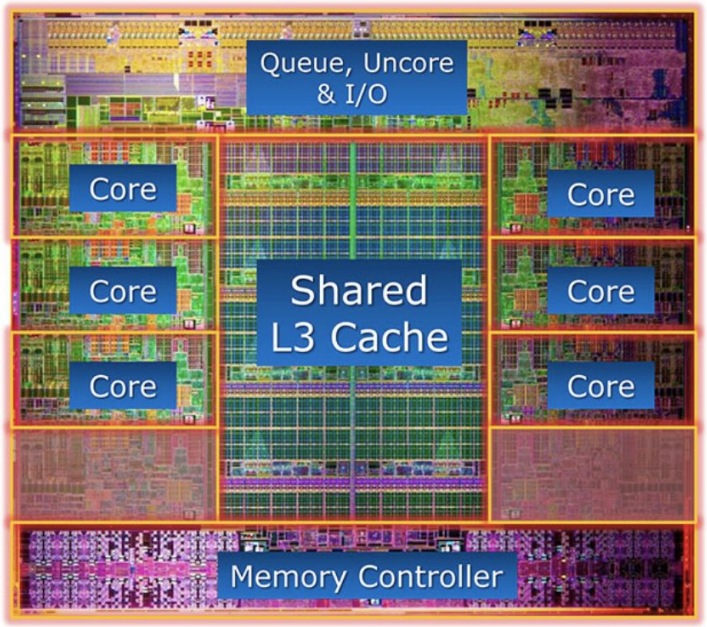
(prices from 2018)

Philipp Koehn;
https://www.cs.jhu.edu/~phi/csf/slides/lecture-cache.pdf

# OK, so what's our strategy?

- Really big caches? **Where would we put them?**

Intel® Core™ i7-3960X Processor Die Detail



# OK, so what's our strategy?

- We want:  lots of memory and access it fast
- We really have:  different speed/size tradeoffs
- Need methods to give illusion of large and fast memory

Philipp Koehn;
https://www.cs.jhu.edu/~phi/csf/slides/lecture-cache.pdf

# Cache

- CPU needs data at address.

- Is data in cache?
  - Yes? Great. **"Cache hit"**
  - No? Fetch data from main memory, place into cache. **"Cache miss"**
    - If cache is full, evict something already in cache.

# Locality saves the day

- Temporal locality
  - The concept that a resource that is referenced at one point in time will be referenced again sometime in the near future.
- Spatial locality
  - The concept that likelihood of referencing a resource is higher if a resource near it was just referenced.

Marc Snir;
https://snir.cs.illinois.edu/PDF/Temporal%20and%20Spatial%20Locality.pdf

# Taking Advantage of Locality

- Goal: get performance (speed) of cache, with the capacity (size) of main memory.

- Techniques:
  - **Caches** exploit temporal locality.
  - **Cache Line Size** and **Prefetching** exploit spatial locality.

# Two very similar programs…

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[i][j] = 47;
        }
    }
    return 0;
}
```

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[j][i] = 47;
        }
    }
    return 0;
}
```

Based on example from Philipp Koehn;
https://www.cs.jhu.edu/~phi/csf/slides/lecture-cache.pdf

# Two very similar programs…

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[i][j] = 47;
        }
    }
    return 0;
}
```

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[j][i] = 47;
        }
    }
    return 0;
}
```

Based on example from Philipp Koehn;
https://www.cs.jhu.edu/~phi/csf/slides/lecture-cache.pdf

# Violation of Locality

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[i][j] = 47;
        }
    }
    return 0;
}
```

```
static const int SIZE = 32768;
int matrix[SIZE][SIZE];

int main() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            matrix[j][i] = 47;
        }
    }
    return 0;
}
```

```
√ ~/coding/cs107e_ta_f21/your_lecture % time ./locality
./locality  2.55s user 1.22s system 99% cpu 3.790 total
```

```
√ ~/coding/cs107e_ta_f21/your_lecture % time ./nonlocality
./nonlocality  22.42s user 1.43s system 98% cpu 24.323 total
```

**2.55 seconds**

**22.42 seconds**

Based on example from Philipp Koehn;
https://www.cs.jhu.edu/~phi/csf/slides/lecture-cache.pdf

# Making the Most of "Downtime"

- Despite our best efforts, CPU will need to wait for main memory (or worse, disk or network I/O!)

- Techniques:
  - **Multithreading**
  - **Speculative Execution**

# Intermission: Internships

- https://forum.stanford.edu/
  Resume reviews, mock interviews, hosted tech talks, career fairs
- Ask friends for internal referrals
- If you are interested in your professor's research field, ask them what companies exist in the space that you may be able to reach out to.
- In my experience, recruiting is a black box.
- Cracking the Coding Interview
- CURIS, REU, GEI, SGS

# A Whirlwind Tour

Exposure to important topics…
…with some fun intermissions

- Memory Hierarchy
- **Trust and Safety**
- Memory Safety
- Where to Go From Here

# Trust and Safety

Cybersecurity: Protect against technical attacks that make systems do things they are not designed to do.

- e.g. computer viruses

# Trust and Safety

Cybersecurity: Protect against technical attacks that make systems do things they are not designed to do.

- e.g. computer viruses

Trust and Safety: Protect against abuse – systems doing what they were designed to do, but used in a way that harms people.

## Unique Aspects of Trust and Safety

1. The study of how people abuse the internet to cause harm.
2. Often using products the way they are designed to work.
3. Crosses between specialties. Requires understanding of society and humanity.
4. Is dynamic and unpredictable.

# Trust and Safety

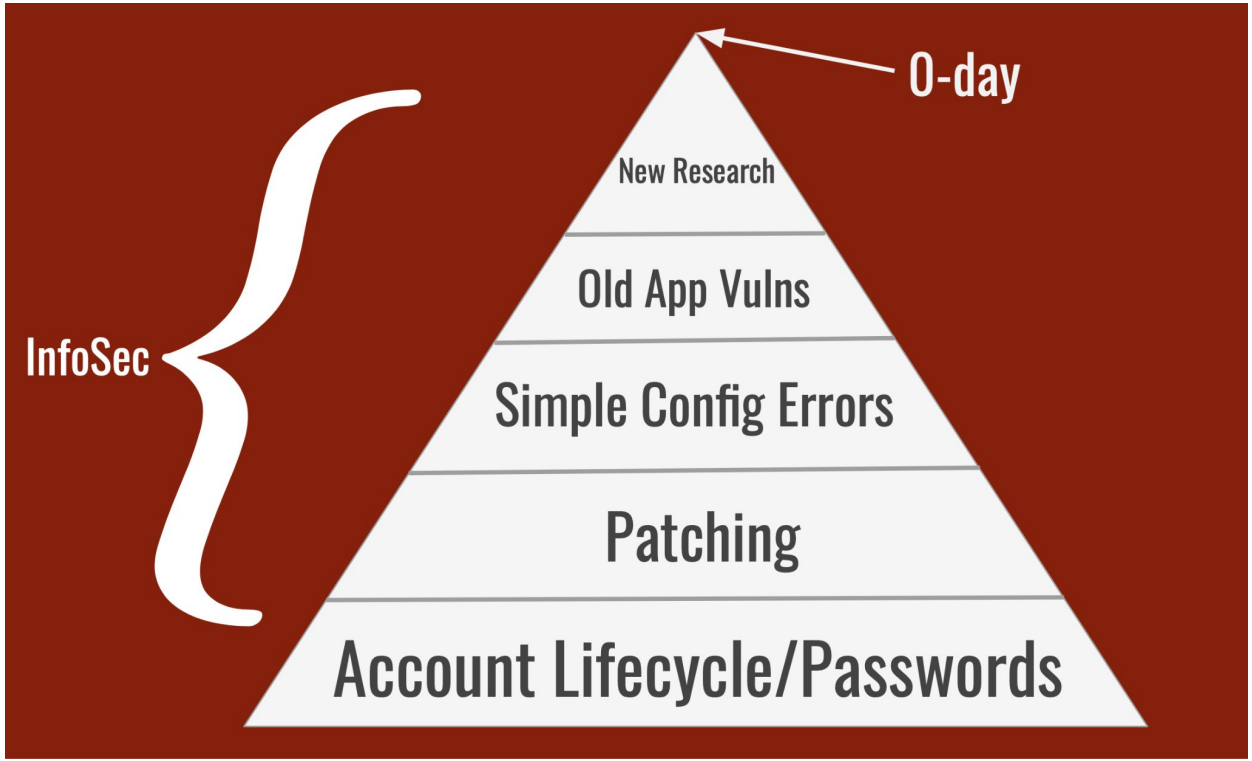
Protect against people using systems *as designed*, but in ways that harm people.

e.g. social media platforms ~~could be~~ **are** used to:

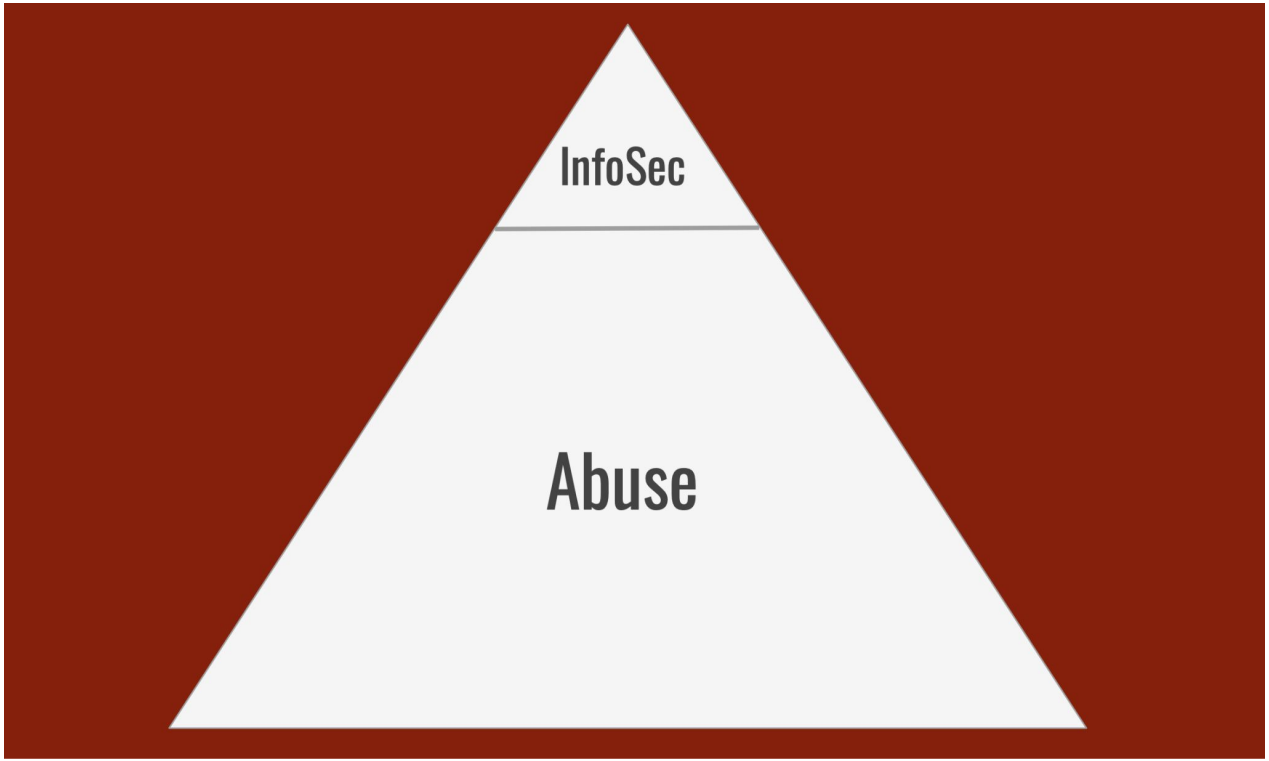
- Spread disinformation (ask me if interested in more)
  - From political influence campaigns…
  - …to enabling genocide in Myanmar
- Coordinate harassment/violence against individuals
  - Doxxing
- Recruit and radicalize violent extremists
- Disseminate CSAM, conduct sextortion

# Trust and Safety: Motivation

# Trust and Safety: Motivation



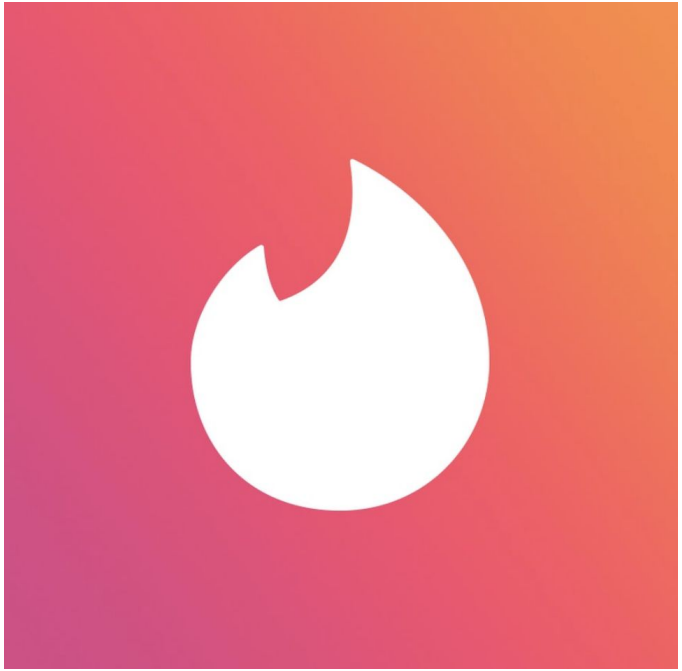
# Trust and Safety: Motivation



# Trust and Safety: What can we do?

- Understand common ways technologies (esp. Internet tech) are used to cause harm.
- **Consider Trust and Safety concerns when designing and building your products.** Think through impact of new tech on users and other groups.
- Design user interfaces and features to minimize abuse potential.
- Provide support channels (e.g. reporting) to address abuse when it appears.
- Have empathy for a broad cross-section of the people who use your products and the risks they face.

# Trust and Safety: Case Study



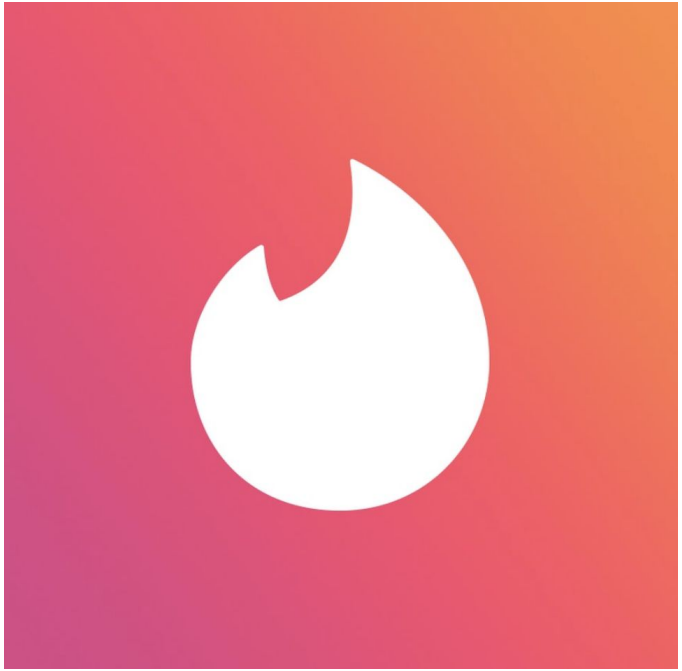
**Feature:** Display distance from your match.

Implementation: Their app **sends their GPS location to your phone**, your app computes the distance, displays rounded to the nearest mile.

Problematic?

https://robertheaton.com/2018/07/09/how-tinder-keeps-your-location-a-bit-private/

# Trust and Safety: Case Study



**Feature:** Display distance from your match.

Implementation: Their app **sends their GPS location to Tinder server**, your app does the same, server computes distance and sends distance to phone.

Problematic?

https://robertheaton.com/2018/07/09/how-tinder-keeps-your-location-a-bit-private/

# Trust and Safety: Case Study


Problematic? Server sends distance with **15 decimal places of precision…**

GPS spoofing of your phone's location
+ trilateration =
30m accuracy of your match's location!

Eventually fixed with rounding distances, so trilateration only works to ~1mile accuracy.

https://robertheaton.com/2018/07/09/how-tinder-keeps-your-location-a-bit-private/

# Trust and Safety: Case Study

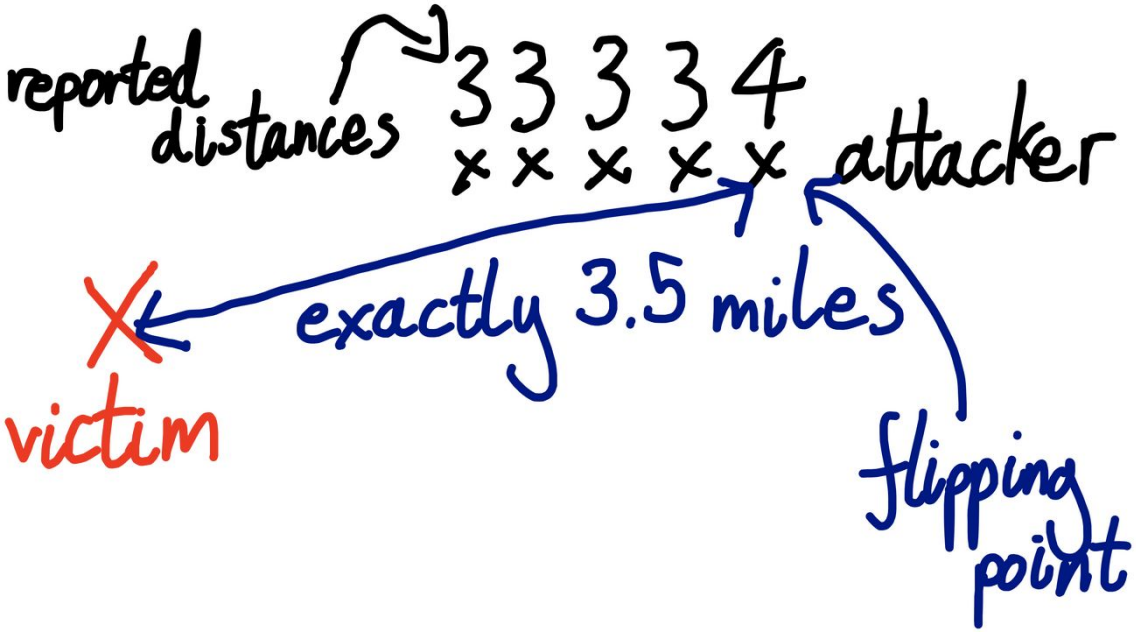

**Feature:** Display distance from your match.

**Implementation:** Distances sent to server, server sends back rounded response.

Problematic?

https://robertheaton.com/bumble-vulnerability/

# Trust and Safety: Case Study




https://robertheaton.com/bumble-vulnerability/

# Responsible Disclosure

## Disclosure timeline

- 2021-06-15: vulnerability reported to Bumble via HackerOne
- 2021-06-18: Bumble deploy a fix
- 2021-07-21: disclosure agreed via HackerOne

https://robertheaton.com/bumble-vulnerability/

# Trust and Safety: Another Example

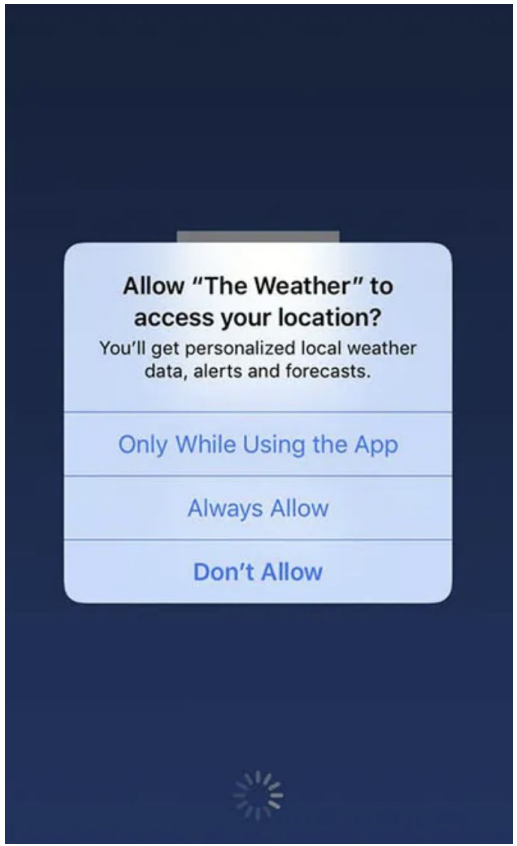


https://www.nytimes.com/2022/02/11/technology/airtags-gps-surveillance.html

https://support.apple.com/en-us/HT212227

# Violations of Trust and Safety

The Weather Channel




https://www.technologyreview.com/2019/01/07/1517/the-weather-channel-app-has-been-accused-of-tracking-users-and-then-selling/

https://9to5mac.com/2020/08/20/ibm-settles-la-lawsuit-over-the-weather-channel-app-selling-user-location-data/

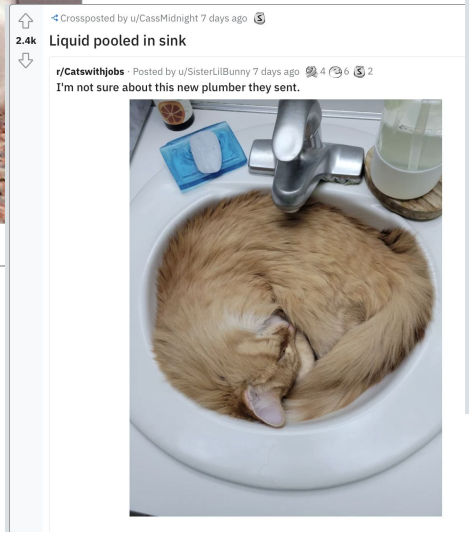
# Violations of Trust and Safety

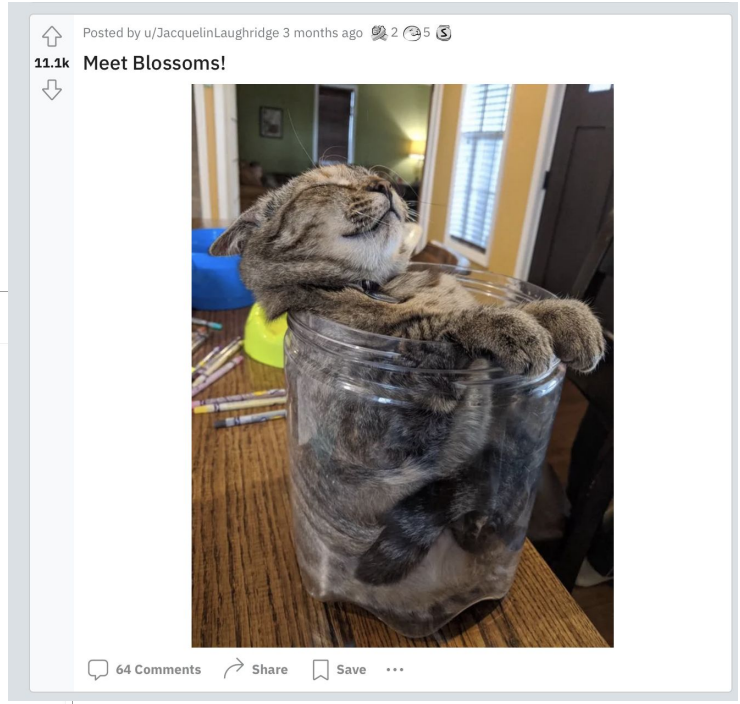
- **Gross Negligence:** Equifax (patching), Solarwinds (passwords)
- **Fraud:** Theranos
  - Including lying to patients!
- **Clearly Unethical Behavior:** Uber
  - “Greyball”
  - “Hell”
- **Aggressive Design, Ignoring Warnings:** Zoom

https://www.cnet.com/tech/tech-industry/uber-to-end-use-of-greyball-tool-to-evade-officials/
https://www.engadget.com/2017-09-08-uber-federal-investigation-hell-program.html
Zoom: https://www.youtube.com/watch?v=K7hIrw1BUck

# Trust and Safety: Stanford Case Study


# Trust and Safety: Humans are Cats


r/catsareliquid

# Trust and Safety: Culture

Behaviors we Tolerate

**Culture**

Behaviors we do not Tolerate

# A Whirlwind Tour

Exposure to important topics

- Memory Hierarchy
- Trust and Safety
- **Memory Safety**
- Where to Go From Here

# Memory Safety

Memory Safety Errors include:

- Buffer overflows
- Double free
- Use-after-free
- Not checking malloc() for NULL
- Using uninitialized  memory
- and more…

# Memory Safety

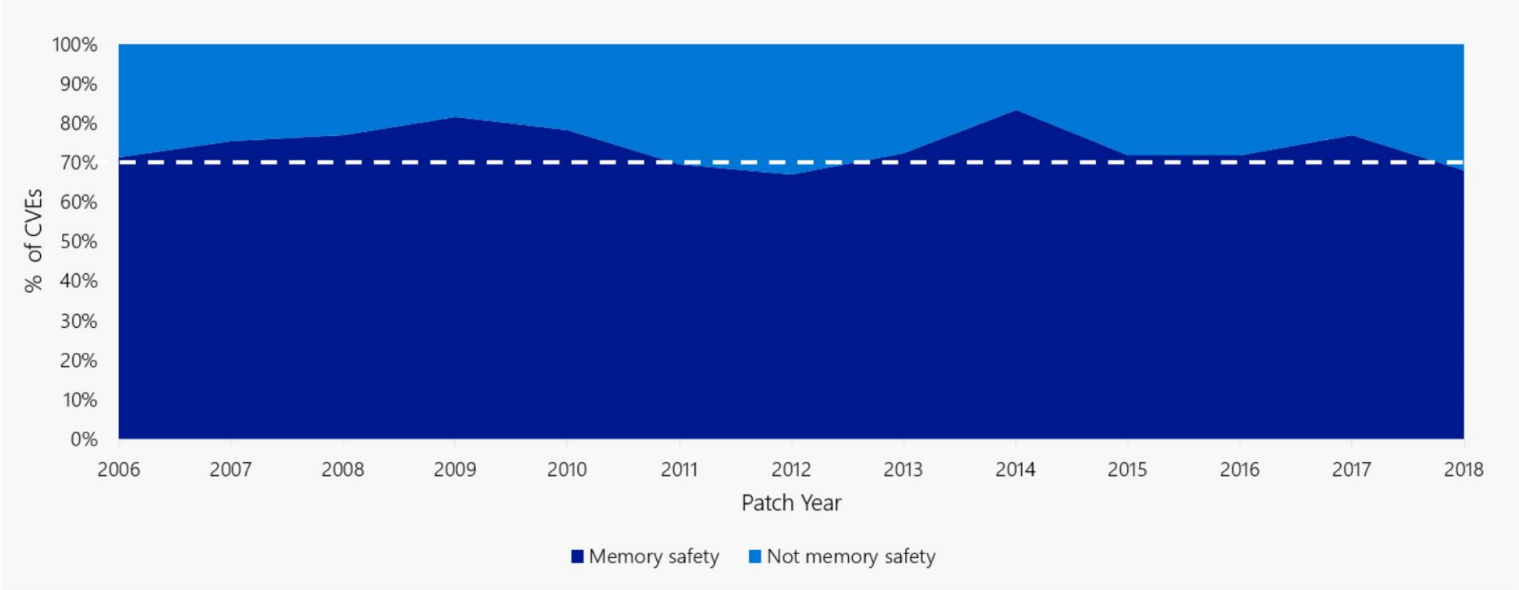



Figure 1: ~70% of the vulnerabilities Microsoft assigns a CVE each year continue to be memory safety issues

https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/
https://msrc-blog.microsoft.com/2019/07/18/we-need-a-safer-systems-programming-language/

# Memory Safety

**Banned** functions in git codebase:

```
11  #define BANNED(func) sorry_##func##_is_a_banned_function
12
13  #undef strcpy
14  #define strcpy(x,y) BANNED(strcpy)
15  #undef strcat
16  #define strcat(x,y) BANNED(strcat)
17  #undef strncpy
18  #define strncpy(x,y,n) BANNED(strncpy)
19  #undef strncat
20  #define strncat(x,y,n) BANNED(strncat)
21
22  #undef sprintf
23  #undef vsprintf
24  #ifdef HAVE_VARIADIC_MACROS
25  #define sprintf(...) BANNED(sprintf)
26  #define vsprintf(...) BANNED(vsprintf)
27  #else
28  #define sprintf(buf,fmt,arg) BANNED(sprintf)
29  #define vsprintf(buf,fmt,arg) BANNED(vsprintf)
30  #endif
31
```

https://github.com/git/git/blob/master/banned.h

https://devblogs.microsoft.com/oldnewthing/20050107-00/?p=36773

# Memory Safety

```
GETS(3)                          Linux Programmer's Manual                          GETS(3)

NAME
       gets - get a string from standard input (DEPRECATED)

SYNOPSIS
       #include <stdio.h>

       char *gets(char *s);

DESCRIPTION
       Never use this function.

       gets()  reads  a line from stdin into the buffer pointed to by s until either a terminating newline or
       EOF, which it replaces with a null byte ('\0').  No check for buffer overrun is  performed  (see  BUGS
       below).

RETURN VALUE
       gets()  returns  s  on  success, and NULL on error or when end of file occurs while no characters have
       been read.  However, given the lack of buffer overrun checking, there can be no  guarantees  that  the
       function will even return.
```

# A note about Rust…

Rust offers C/C++ level performance with powerful compile-time memory safety checks.


https://stackoverflow.com/questions/36136201/how-does-rust-guarantee-memory-safety-and-prevent-segfaults

https://stanford-cs242.github.io/f18/lectures/05-1-rust-memory-safety.html

https://www.rust-lang.org/

# Intermission: Sneakernet

# Intermission: Sneakernet

*Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.*
*–Andrew Tanenbaum, 1981*

https://what-if.xkcd.com/31/

# Intermission: Sneakernet

"We have a number of machines about the size of brick blocks, filled with hard drives.

"We send them out to people who copy the data on them and ship them back to us. We dump them on to one of our data systems and ship it out to people."

–informal Google program, ca. 2007

http://news.bbc.co.uk/2/hi/technology/6425975.stm

# Intermission: Sneakernet


AWS Snowmobile is an Exabyte-scale data transfer service used to move extremely large amounts of data to AWS. You can transfer up to 100PB per Snowmobile, a 45-foot long ruggedized shipping container, pulled by a semi-trailer truck. Snowmobile makes it easy to move massive volumes of data to the cloud, including video libraries, image repositories, or even a complete data center migration. Transferring data with Snowmobile is more secure, fast and cost effective.

https://aws.amazon.com/snowmobile/

# Intermission: Netflix Chaos Monkey

https://github.com/netflix/chaosmonkey

Chaos Monkey **randomly terminates virtual machine instances and containers that run inside of your production environment**. Exposing engineers to failures more frequently incentivizes them to build resilient services.

# A Whirlwind Tour

Exposure to important topics…
…with some fun intermissions

- Memory Hierarchy
- Trust and Safety
- Memory Safety
- **Where to Go from Here**

# Life after CS107e: Where to Go From Here

Systems Core:

- CS110/111
- CS112/140
- CS140E
- CS240/CS240LX

Systems:

- CS144 Networking
- CS149 Parallel Computing

Ethics:

- CS182

# Life after CS107e: Where to Go From Here

Other CS:

- **CS166** Data Structure
- CS242 Programming Languages (Rust!)

Computer Architecture:

- EE108 Digital Systems Design
- EE180 Digital Systems Architecture
- **EE282** Computer Systems Architecture

Circuits/Hardware:

- **ENGR40M** Intro to Making
- EE101A Circuits I

# Life after CS107e: Where to Go From Here

Security:

- **CS155** Computer and Network Security (technical)
- INTLPOL268 Hack Lab (practical exposure + legal)
- CS255 Cryptography (technical)
- CS356 Topics in Computer and Network Security (papers + project)
- **CS152** Trust and Safety Engineering

International Security:

- **MS&E193** Technology and National Security
- **POLISCI114S** International Security in a Changing World
- CISAC talks, etc.

# Life after CS107e: Where to Go From Here

Talk to your friendly course staff!

>> stay tuned for signups for one-on-one course reflection meetings

# My favorite Bug (anecdotal)