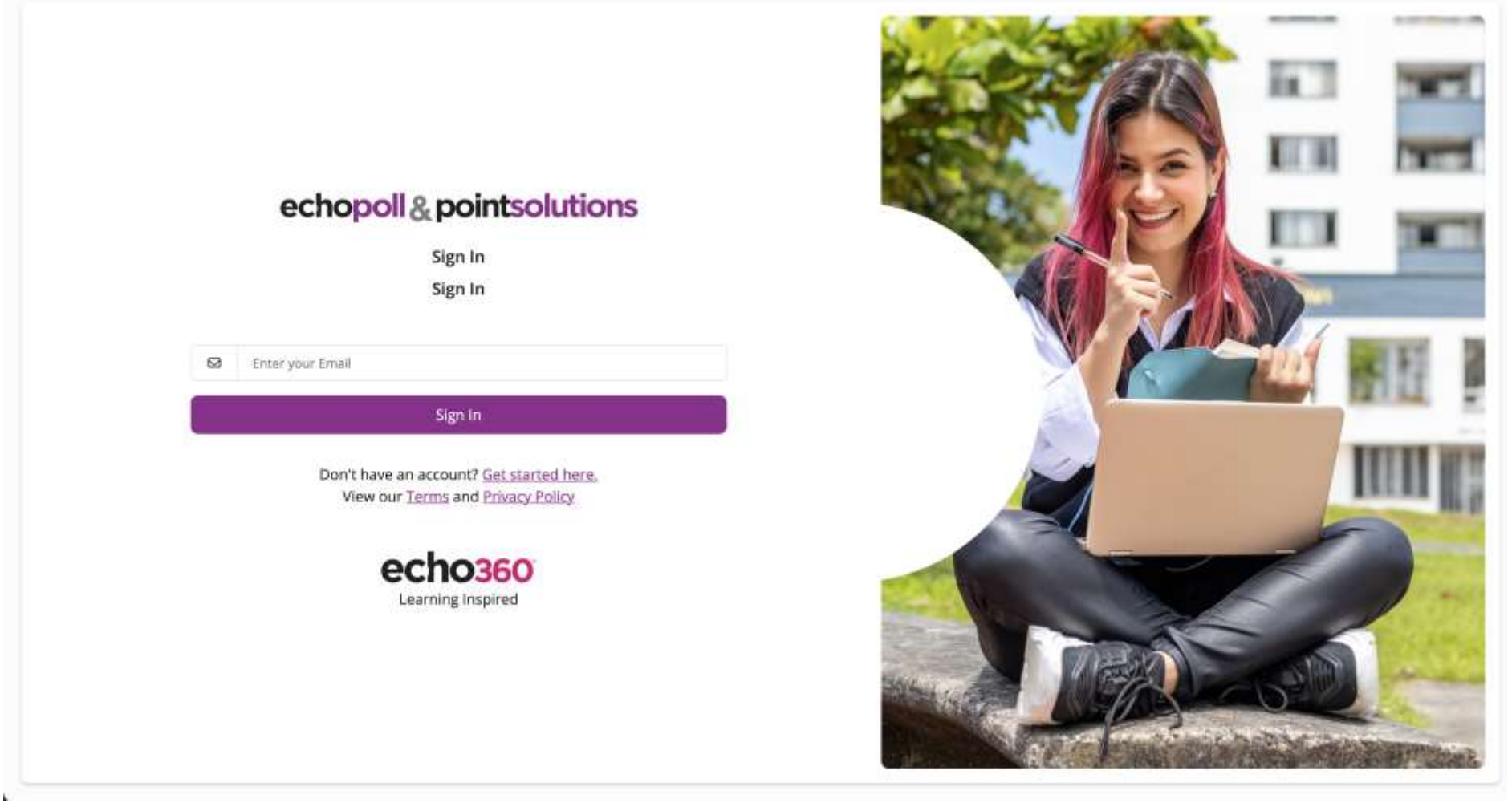
# COSC 366
# Intro to Cybersecurity

Dr. Suya

Fall 2024

# Daily Updates

- Late policy:
  - 5 free days, after that, 10% of the grade will be deducted for each late day.
  - If you submit early: get additional free days; once accumulated free days exceed a threshold, you get bonus points
- Plan to use PointSolution to track attendance
  - https://echo360.com/get-started-with-point-solutions/
  - No penalty if you miss the class
  - Get 1 point if you answer questions, and 2 points if you answer correctly
  - The questions disappear after 2 minutes
  - Check the instruction in the announcement later


echopoll & pointsolutions
Sign In
Sign In
Enter your Email
Sign In
Don't have an account? Get started here.
View our Terms and Privacy Policy
echo360
Learning Inspired

# Daily Updates

- TA office hours:
  - Mon 11:30-12:30, Wed 12:30-1:30, Fri 1:30-2:30
  - Location: MK 235
- Instructor office hour
  - Tue 10:30-11:30
  - Location: MK 344

# Request from a colleague

- Prof. Fei Liu
  - Research in robotics, seeking undergraduate researchers
  - Teaching reinforcement learning (ECE 414/517) this semester, can audit the class if interested
  - Webpage: https://lnnx2006.github.io
  - Email: fliu33@utk.edu

# Today's Class

- Security concepts 101
- Think like an adversary

# What Is Cybersecurity?

- Systems without security
- Systems with security
- The key difference:
  - Security involves an adversary who is malicious, active and dynamic
  - Attackers constantly seek to circumvent protective measures

# What Is Cybersecurity?

- Attackers are not normal users
- Normal users: try to avoid bugs/flaws
- Attackers: try to find the bugs/flaws out and to exploit them

# What Does It Mean to be Secure?

- There is no such thing as security, only degrees of insecurity
- Goal: raise the bar for the attacker
  - Too difficult
  - Too expensive
- Ultimately, we want to mitigate undesired behavior while maintaining main functionality

# What Are **Undesired** Behaviors?

- Reveals information users want to hide: **Confidentiality**

# Confidentiality

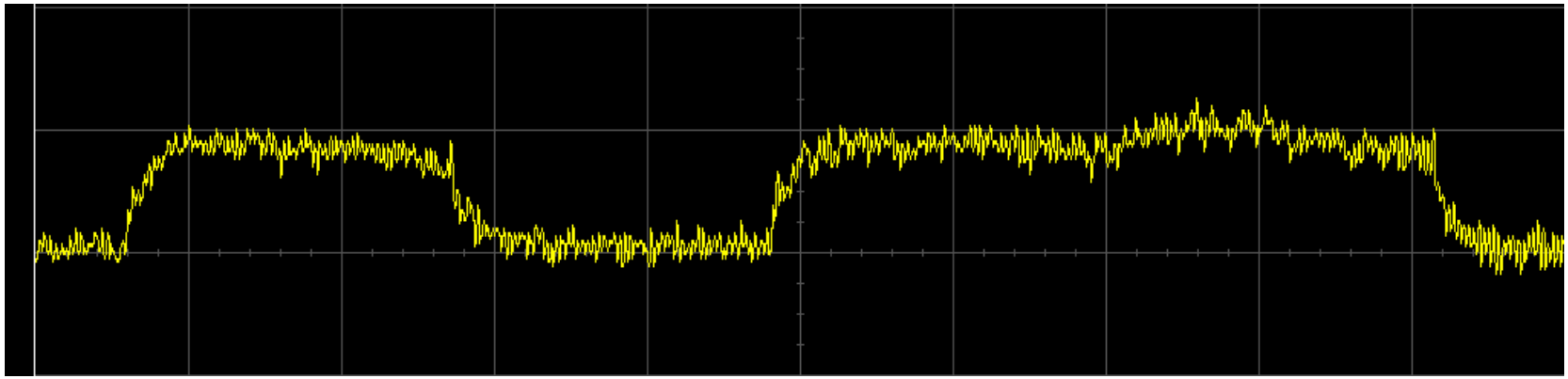
- Only authorized parties should learn certain information
- Examples:
  - The contents of a file on a disk, a value in a database, etc.
  - Information traveling between two parties over a network
  - The fact that two parties are communicating (anonymity and privacy)

# Confidentiality Challenges

- Need to think carefully and clearly define
  - What data is protected
    - Just because *some* data is confidential in a system, it does not mean *all* data needs to be confidential
  - Who is authorized
- Need to think about ***side channels*** – flaws that are in the implementation, not the design

# Side channel attacks

- Power analysis
  - Power consumption of devices
  - Different operations or values may have different patterns



# Side channel attacks

- Any other side channels?

# Side channel attacks

- Acoustic
  - Sound emission of devices
  - Examples: key presses, hard-drive noises, printer noises
- Heat
  - Heat emissions from devices such as CPU, GPU
  - Certain operations require more processing (more heat)
- Cache access
  - Difference in cache accesses or timing differences

**Becomes more vulnerable with the advancement of ML models!**

# What Are **Undesired** Behaviors?

- Reveals information users want to hide: **Confidentiality**

- Modified information or functionality: **integrity**

# Integrity

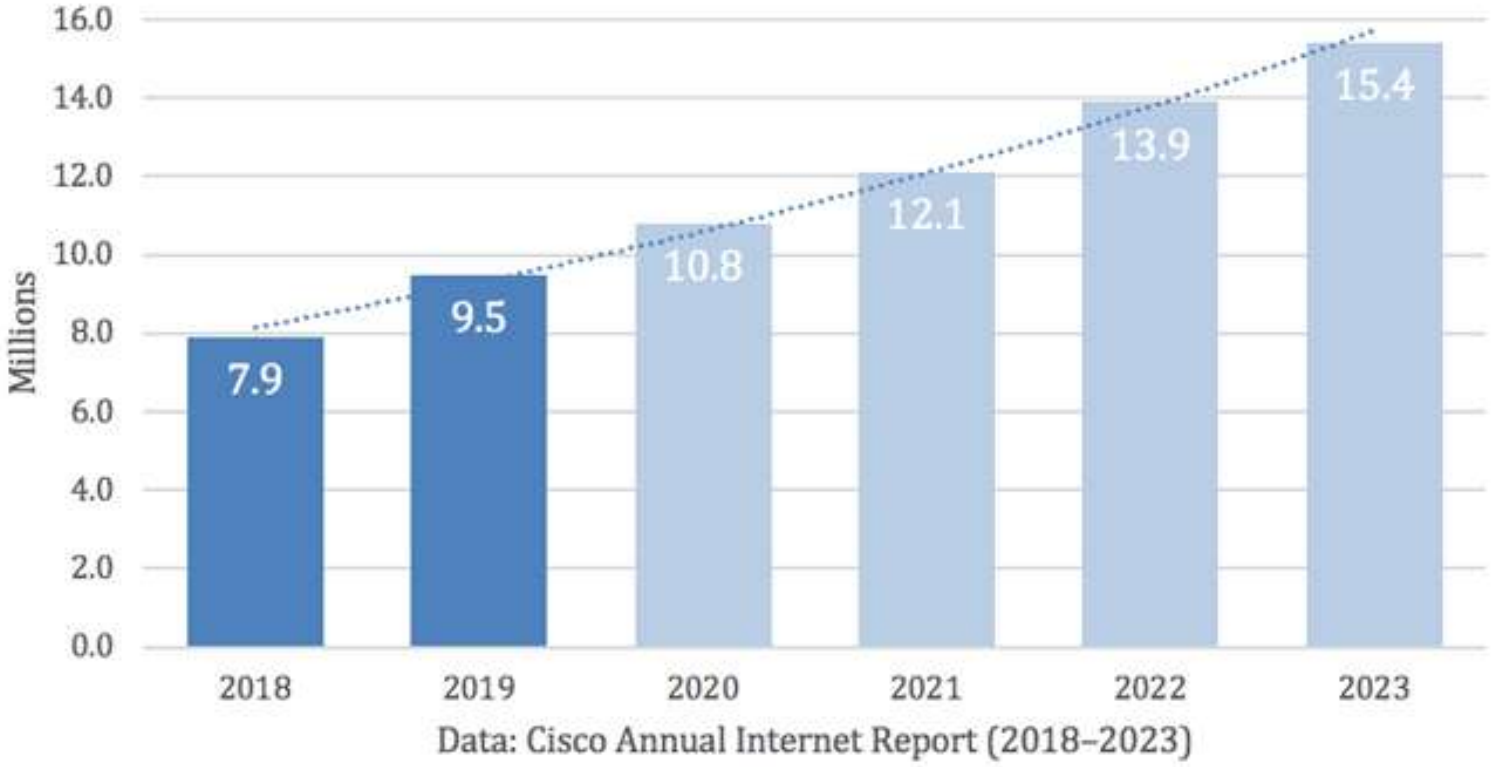
- Everything is as it should be
- Examples:
  - Data integrity - Only an allowed party can write to a particular resource
  - Authentication integrity - An entity should be who they claim to be
  - Computational integrity - A function should correctly compute a result
  - Business logic integrity - A computation should result in the intended behavior under application specifics requirements (e.g., amazon discounts)
  - Control flow integrity - A function should do the task it was written for

# What Are **Undesired** Behaviors?

- Reveals information users want to hide: **confidentiality**
- Modified information or functionality: **integrity**
- Denies access to a service: **availability**

# Availability

- "Authorized users" should be able to interact with resources when they wish to in the



# More on the CIA Triad

- CIA Triad: Confidentiality, Integrity, Availability
- Most efforts spent on confidentiality historically
- Integrity and availability are more important in emerging and critical applications
- Examples: self-driving cars, drones, power grid, missile defense system
- Machine learning security:
  - Three undesired behaviors are still present, but many do not clearly categorize or recognize them.

# Golden Properties (of Secure Systems)

- Authentication
  - Proof that an entity is/owns/controls an identity

# Golden Properties (of Secure Systems)

- Authorization
  - Ensure that an entity has the necessary privilege to take the requested action
  - Least privilege: grants users and processes only the permissions necessary for their tasks, minimizing the risk of unauthorized access.

# Golden Properties (of Secure Systems)

- Accountability/auditability
  - Ability to link (past) actions to the entity that executed them
  - You should have a clear view of who is responsible for a given behavior
  - Having good visibility into your system and its behavior is as important as building a secure system

# Defining "Secure" Systems

- System designers must choose what subset of these goals are important to their system

- Secure systems need *not* ensure *all* security goals are achieved
  - Can have significantly high cost, impossible to achieve

# Defining "Secure" systems

- Security Control
  - Method by which security goals are achieved, e.g., encryption
  - Essentially the opposite of an "Attack"
- Security Mechanism
  - Implementation of a security mechanism, e.g., AES, RSA

# Defining "Secure" systems

- Security mechanisms have a cost
  - Processing time, storage space, programmer time, code complexity, pushback in the real world, etc.
  - This why you do not always try to defend against EVERYTHING
  - **Spend your (limited) security resources wisely**

# Attacks

- An attack undermines one or more security goals
  - An adversary is an entity that is implementing and/or launching attacks
  - A vulnerability can be exploited by an adversary to launch attacks

# Attacks

- Attacks result generally from:
  - Mismatches in "mental models" and "actualized models" of systems
  - Unfounded assumptions: legit users only
  - Unenforced assumptions: user should use complex passwords
  - Misplaced trust: trust in third-party software without validation (SolarWinds Orion Hack)
  - Feature Creep: gradually add features that are not secure

# Know Your Adversary

- Different adversaries have different capabilities
  - A script-kiddie can …
  - A nation state can …
- Define ***who*** your adversary is, build mechanisms against them
- **Threat Model**: clearly defines adversary knowledge, adversary capability

# Know Your Adversary

- Some adversaries are more likely than others...
  - Odds a random script-kiddie wants my password?
  - Odds the NSA wants my password?
- The stronger the adversary, often the more costly the security mechanisms become
  - Again, spend your security resources wisely!

# Why Do Attackers Do This?

- A lot of reasons…
  - To make money
  - To cost you money
  - To make money by costing you money
  - Because they are a terrorist
  - Because they are a freedom fighter
  - Because they are a government, and you are a different one
  - Because they are at war with you
  - Because it is cool
  - Because they are mad
  - Because they can

# Building An Attack

- Adversaries (generally) approach this process by keeping in mind
  - Their end goal
  - What unfounded/unenforced assumptions, mistakes, bugs they have found
- Each assumption by itself might not ***directly*** lead to undermining a security property
  - But one bug might lead to the ability to create another….
    - SQL injection -> browse database to find other vulner.
- The challenge for an attacker is to find these little flaws, and chain them together into something bigger….

```
// Checks if there is enough money in the account to handle the charge, and if so,
executes the charge
public boolean chargeAccount(double debitAmount) {
    if (this.balance >= debitAmount) {
        this.balance -= debitAmount;
        return true;
    } else {
        return false;
    }
}
```

What is the assumption?
What can an adversary do with it?

```
// Checks if there is enough money in the account to handle the charge, and if so,
executes the charge
public boolean chargeAccount(double debitAmount) {
    if (this.balance >= debitAmount) {
        this.balance -= debitAmount;
        return true;
    } else {
        return false;
    }
}
```

*myself.chargeAccount(-100.00);*

What else?

```
// Fetch information for a single client
public Object executeClientLookup(String name) {
    // Build the SQL query to fetch client information based on the client's name
    String sql = "SELECT * FROM client WHERE name = '" + name + "'";

    // Execute the SQL query using the sqlEngine and return the result
    return this.sqlEngine.execute(sql);
}
```

What is the assumption?
What can an adversary do with it?

‘name’ is directly inserted into query string, attacker can inject arbitrary SQL code

```
// Fetch information for a single client
public Object executeClientLookup(String name) {
    // Build the SQL query to fetch client information based on the client's name
    String sql = "SELECT * FROM client WHERE name = '" + name + "'";

    // Execute the SQL query using the sqlEngine and return the result
    return this.sqlEngine.execute(sql);
}
```

*executeClientLookup(“Bob’ OR 1 = 1 --”);*

```
// Fetch information for a single client
public Object executeClientLookup(String name) {
    // Build the SQL query to fetch client information based on the client's name
    String sql = "SELECT * FROM client WHERE name = '" + name + "'";

    // Execute the SQL query using the sqlEngine and return the result
    return this.sqlEngine.execute(sql);
}
```

*executeClientLookup("Bob'; drop table client --");*

```
// Directory where we keep our HTML files
public final String WEBPAGE_ROOT_DIRECTORY = "/home/webhome/";

// Gets requested page from HTTP request... calls fetchWebpageFile
public byte[] fetchWebpageFile(String webPage) {
    // Build the path to the HTML document
    String fullPath = WEBPAGE_ROOT_DIRECTORY + webPage;

    // Read and return
    InputStream fIn = new FileInputStream(fullPath);
    // ...
}
```

What is the assumption?
What can an adversary do with it?

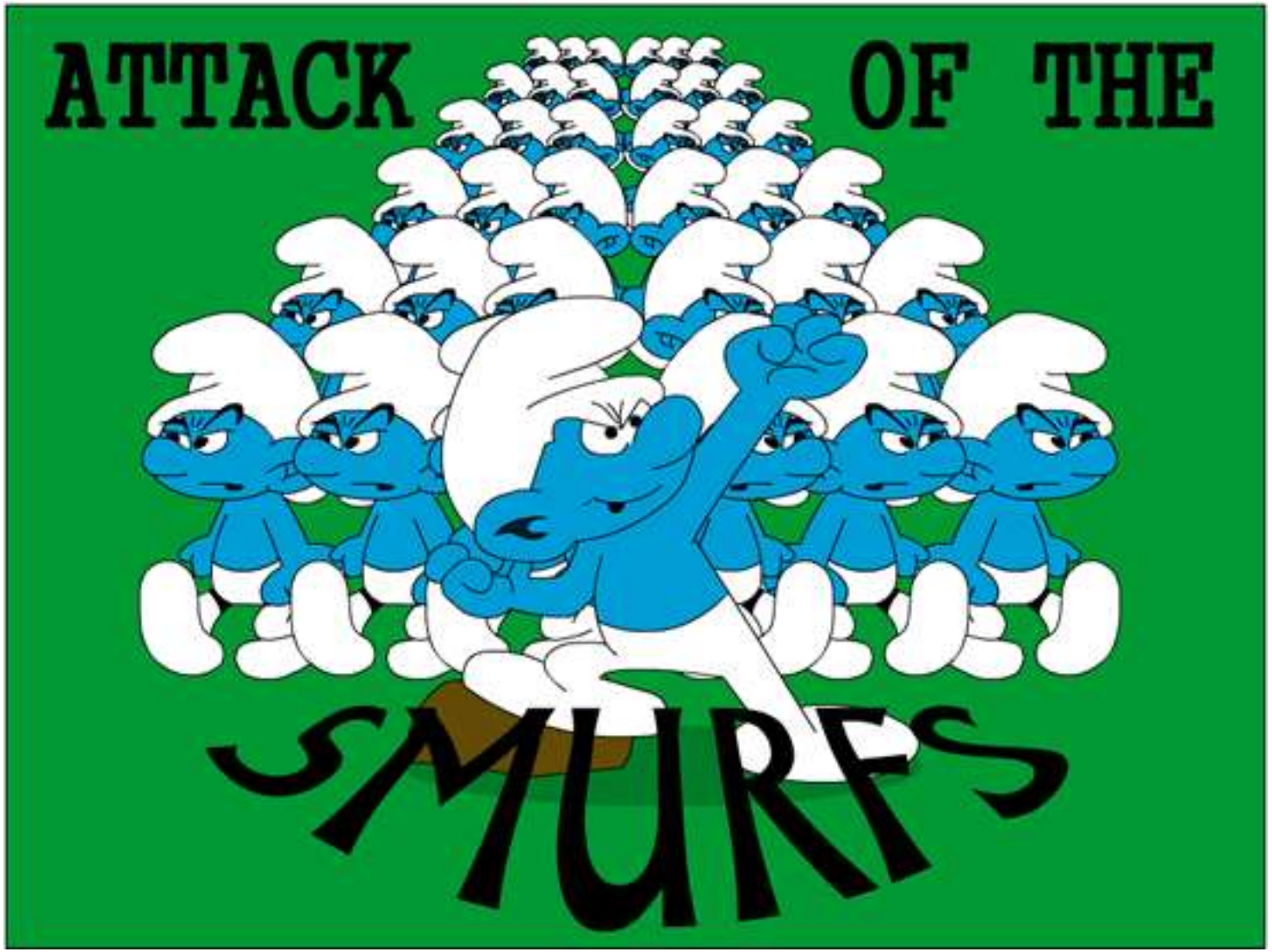
# Directory Traversal Attacks: no sanity check on paths

```
// Directory where we keep our HTML files
public final String WEBPAGE_ROOT_DIRECTORY = "/home/webhome/";

// Gets requested page from HTTP request... calls fetchWebpageFile
public byte[] fetchWebpageFile(String webPage) {
    // Build the path to the HTML document
    String fullPath = WEBPAGE_ROOT_DIRECTORY + webPage;

    // Read and return
    InputStream fIn = new FileInputStream(fullPath);
    // ...
}
```

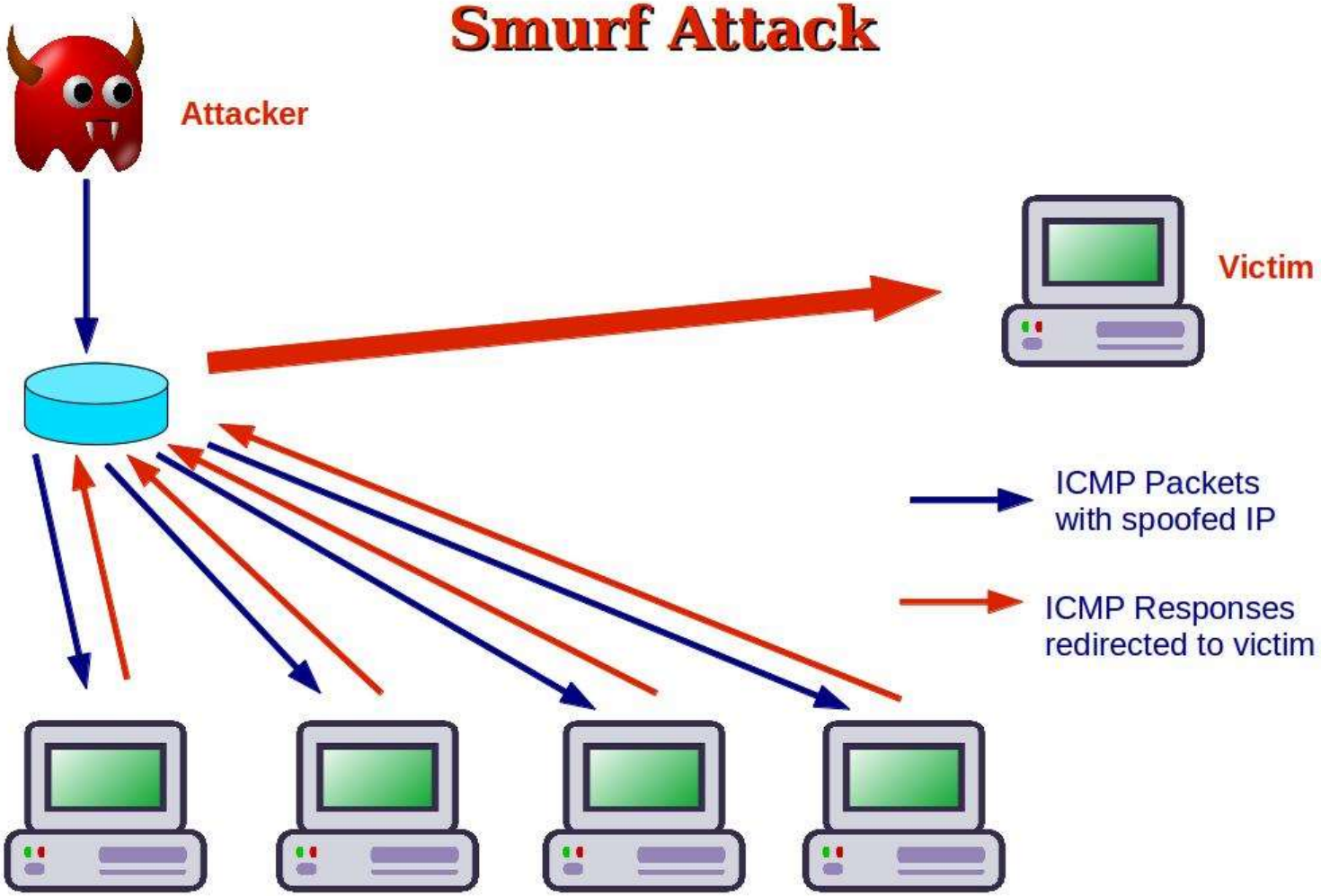
*http://vulnerable.org/nothing/tosee/here/../../../../../etc/shadow*



# Using A Few Assumptions Put Together

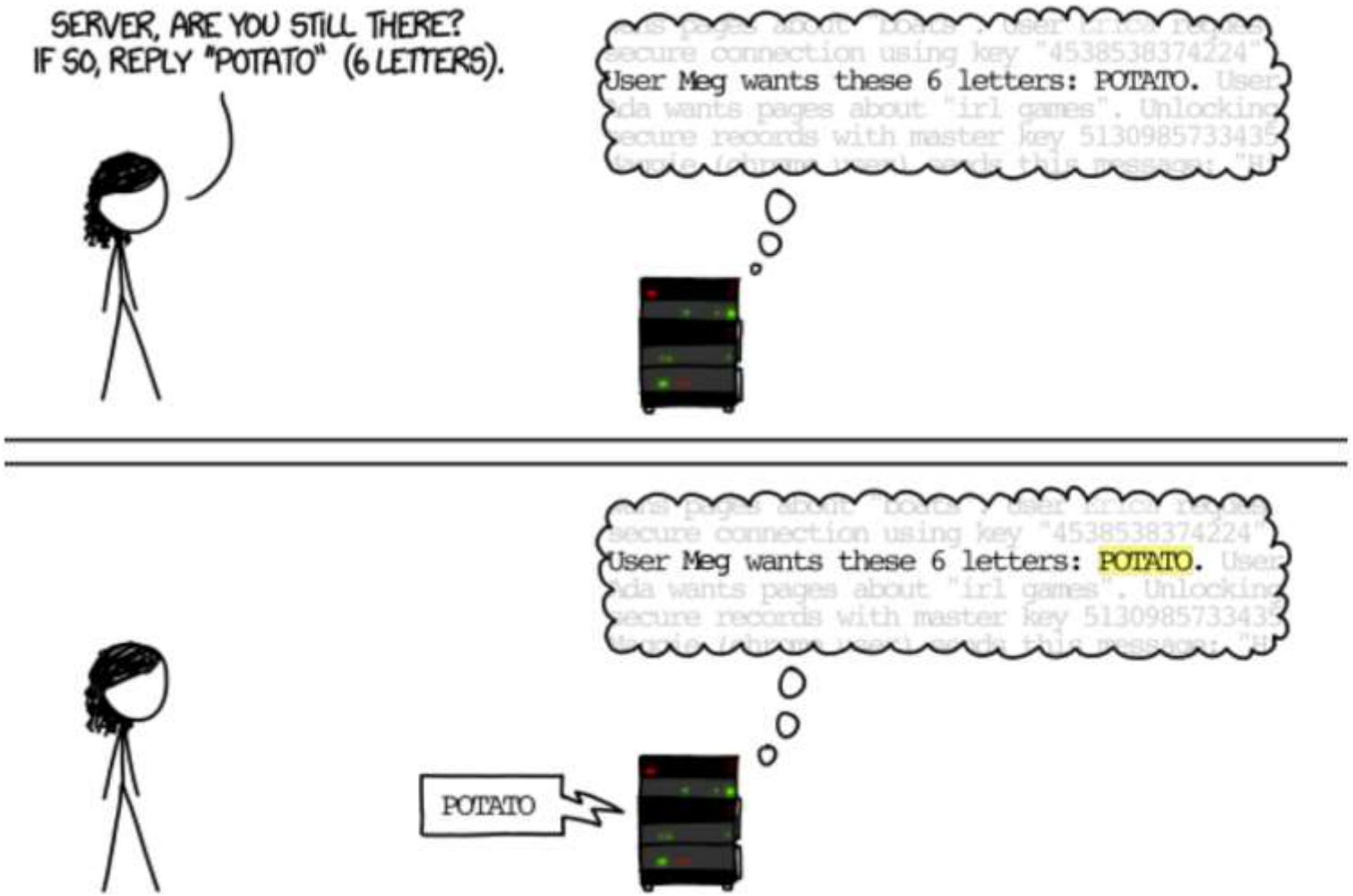
- A protocol exists called ICMP (Internet Control Message Protocol) echo
  - Sender sends a hello message
  - Receiving party sends back a reply message
- There is no integrity on the "from" IP address in a network packet
- Networks have a broadcast address
  - Send a message to that address, it gets broadcast to every device on the network

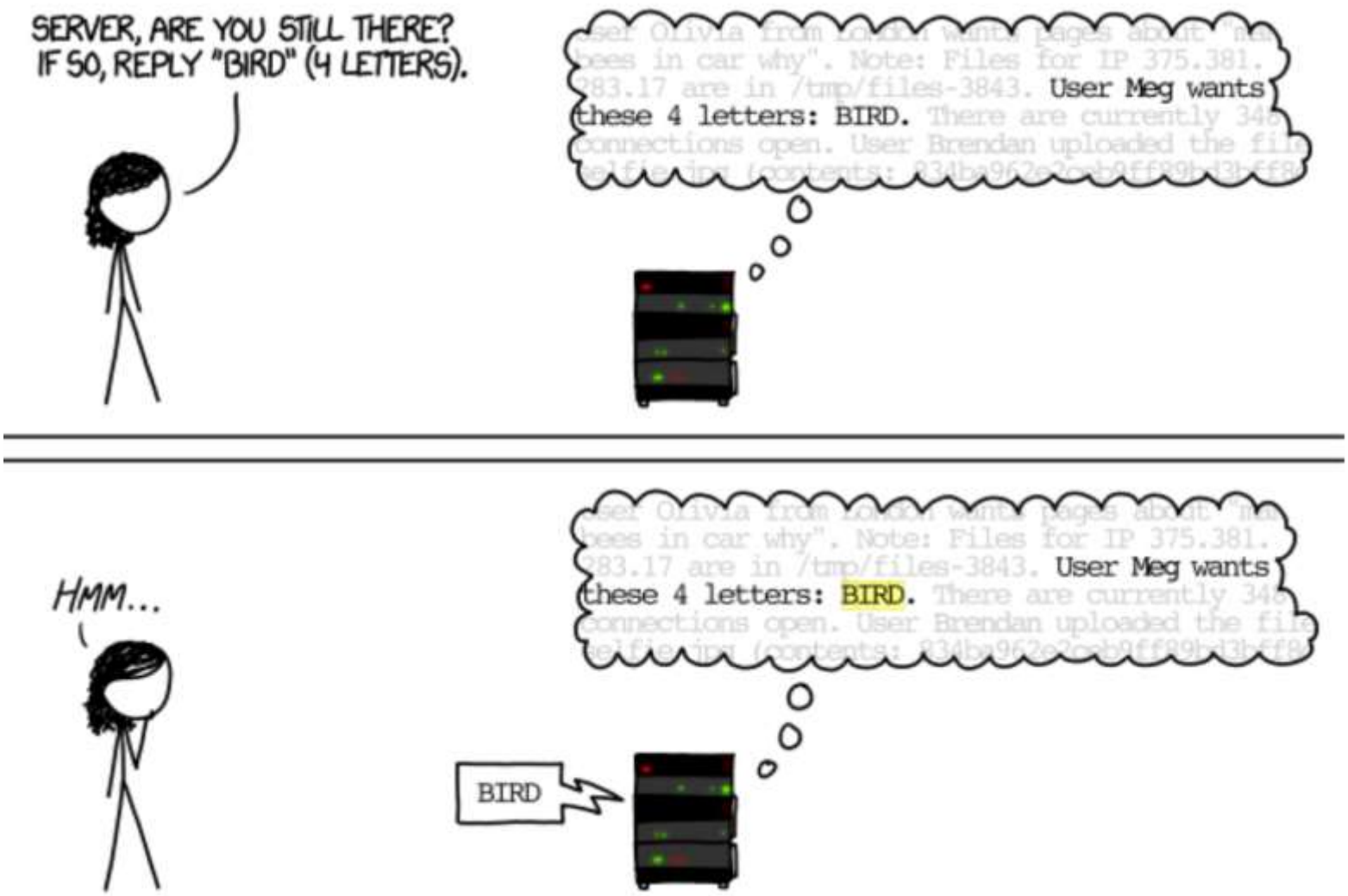
# Smurf Attack

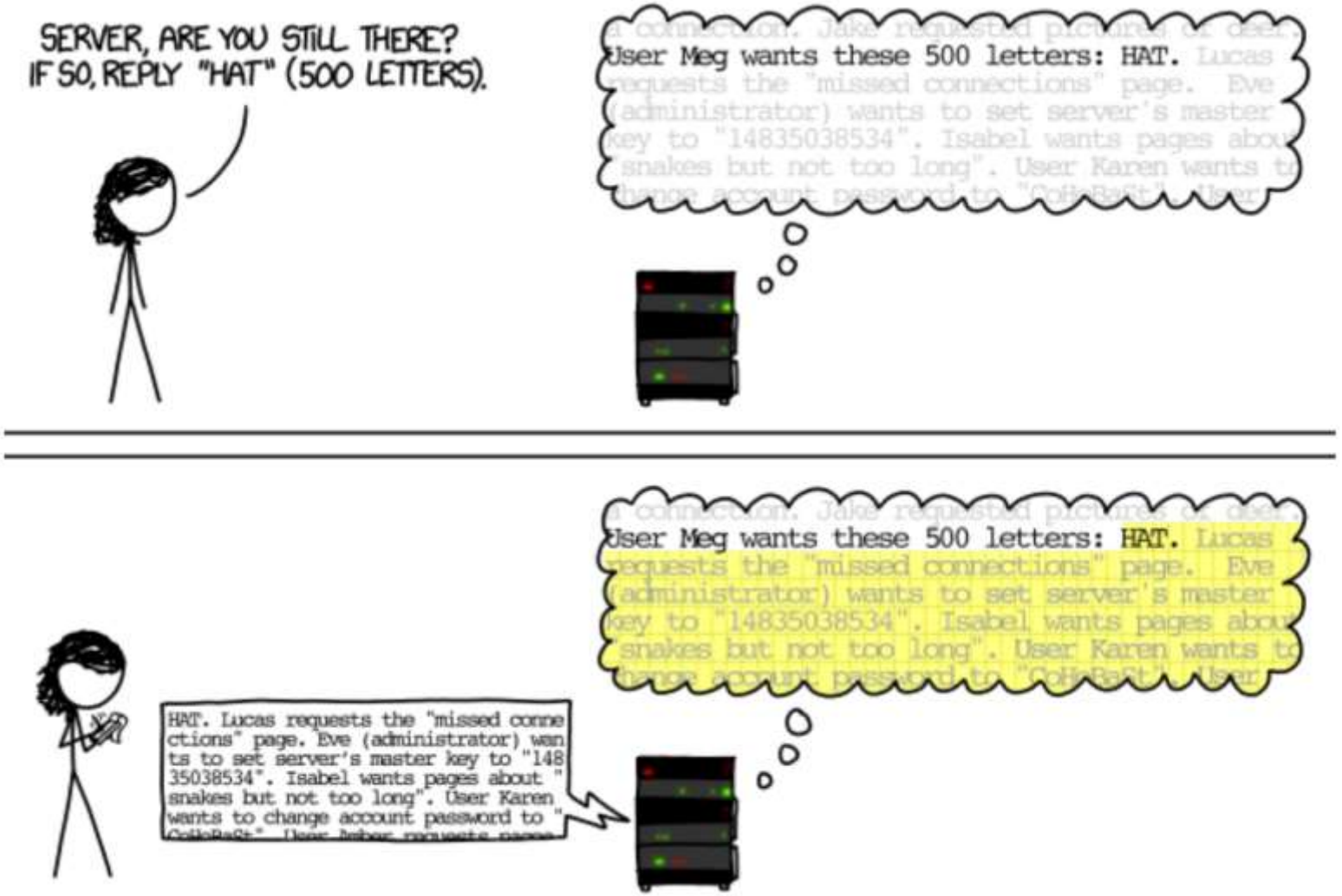


# Heartbleed

- TLS is the de facto protocol for secure online communication
- Heartbleed was a vulnerability in the most popular TLS server
  - A malformed packet allows you to see server memory
- Fix: don't let the user just tell you how much data to give back
- This was a design flaw








- User passwords, private keys, personal information…

~40% of “secure” web servers vulnerable